THE UNIVERSITY OF ALBERTA

A HEURISTIC PROGRAMME FOR
UNIVERSITY TIMETABLING

BY

© FREDERICK TOM ROGERS

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE

OF MASTER OF BUSINESS ADMINISTRATION

FACULTY OF BUSINESS ADMINISTRATION AND COMMERCE

EDMONTON, ALBERTA

FALL, 1972

THE UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research, for acceptance, a thesis entitled A Heuristic Programme for University Timetabling submitted by Frederick Tom Rogers in partial fulfilment of the requirements for the degree of Master of Business Administration.

THE UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research, for acceptance, a thesis entitled [illegible] submitted by [illegible] in partial fulfilment of the requirements for the degree of [illegible].

ABSTRACT

The problem of class-teacher-timetabling for a university department is examined. A description of the present system at the Faculty of Business provides insight into how the problem is being handled. An examination of some of the literature is conducted to determine how the problem is being approached elsewhere. Two methods were found: algorithmic and heuristic. The advantages and disadvantages of both methods are explored. A PL/1 computer program was written to provide the human scheduler with a practical tool for developing a yearly timetable. This program (the FRED system) is a heuristic method of developing a timetable and consists of an interactive terminal system which allows the human to respond to questions from the terminal. The model was then run on a small number of courses to determine limitations and abilities of the system. The program was designed to be "fool proof" and will not abort due to a wrong answer. No prior computer knowledge is necessary to operate this system. A simple terminal session is included to demonstrate some of the capabilities of the system and what questions it asks. Final a program listing and variable dictionary is included.

Acknowledgments

The research done in the thesis could not have been accomplished without the help of many people. In particular I wish to thank: Dr. Peter R. Winters who as supervisor of this thesis contributed many ideas and was patient and understanding of my shortcomings.

Special thanks are also due to Dr. P. Michael Maher, for his support of the work, especially in the early stages and for the many ways he has helped me during my work on my Master's degree.

Dr. Bob Cohoon for the many hours of effort spent in helping me formulate ideas and listening to my ideas; Mr. Ping Wong whose aid in programming was invaluable.

Mr. D.A. McGeachy for his effort in explaining the problem as it exists and providing useful ideas and outcomes on the work for which the result of this thesis are intended to reduce his work load. Mrs Doris Cyre for her typing and formatting the work for the IBM/360 FMT program.

Dr. C. T. L. Janssen for his continued support in the completion of the thesis.

This research was partially supported by an IBM research grant to Dr. Winters.

Finally I wish to thank my parents for their continual

encouragement they provided and specifically my father for his editing and suggestions in preparing this thesis.

Of course all errors found herein are my responsibility.

TABLE OF CONTENTS

ABSTRACT

ACKNOWLEDGMENTS

CHAPTER

1 Introduction 1

The Problem 1

Reasons for Undertaking this Research 11

Aim of the Research 12

2 The Present Timetabling Procedure for the Faculty of Business at the University of Alberta 15

Preparation of the Initial Timetable 15

Advantages of the Present System 18

Disadvantages of the Present System 19

3 State of the Art 21

The Algorithmic Approach 21

Heuristic Models 34

Algorithmic Vs Heuristic 36

Timetabling Work done by A.G. Holtzman 37

4 A Man-Machine Interaction System in a Heuristic Program for University Timetabling: The FRED System 40

The Man's Role in the FRED System 41

The Computer's Role 42

Input to the FRED System 42

The Functions of the FRED System 49

PL/1 as a Source Language 52

5 Results of Application of the FRED System 54

Listing the Timetable 54

Building a Timetable 61

Overall Conclusions 62

Conclusions and Proposed Future Research 63

APPENDIX 1 66

APPENDIX 2 67

LIST OF TABLES

1 Standard Teaching Time Blocks at the
University of Alberta 46

LIST OF FIGURES

FIGURE

1 Request-Matrix on Faculty of Business 1971-72 06

2 Conflicts between Single Section Courses 1971-72 ..09

3 Network for Greko's Model 24

4 Out-of-Kilter Network 26

5 Summary Presentation of Problem 1 with Solution Times 32

6 Summary Presentation of Problem 2 with Solution Times 32

7 Summary Presentation of Problem 3 with Solution Times 33

8 Figurative Representation of Almond Algorithm 34

9 Pre-registration Questionnaire 45

10 Sample List of 1971-72 Timetable 56

11 Sample List of Conflicts 1971-72 57

12 Sample List of 1971-72 Professor Timetables 60

Chapter 1

Introduction

At the University of Alberta the task of developing the academic timetable for the school year is the responsibility of each Faculty with the aid of the Timetabling Officer in the Registrar's Office. The rapid growth of the Business School at the University has made the task of producing the timetable a major effort. In the Faculty of Business Administration and Commerce this procedure has been done entirely by hand and only recently has any attempt[1] been made to computerize this process. Therefore the need for some research into the area and the development of a practical tool was apparent.

The purpose of this chapter is to introduce the problem as established and to outline the objectives of the research.

The Problem

The task of developing a timetable for a university department has become tedious and frustrating work because of the number of variables involved. In simple terms--the problem is to assign professors and courses to time slots; subject to some criteria one of which is to limit the number of conflicts in the students' chosen patterns and to achieve good classroom utilization. It is not our intention to address the question of how the goodness of a timetable is

established. A number of criteria exist, however they would need to include some consideration of student conflicts. These objectives are subject to constraints, some of which are unique to each educational institution.

The following is a list of general constraints which apply to most timetabling problems.

1) No professor can teach more than one class in a given time period.
2) Certain courses must not be taught concurrently to avoid conflicts for students.
3) Certain courses can not be offered in certain time periods. This may be because some room or piece of equipment is not available during these times.
4) There is an arbitrary limit to the number of sections that can be taught concurrently. This is necessary to allow an even distribution of classes throughout the schedule, and reduces the classroom space needed.

One of the most comprehensive examinations of the timetabling problem was done by A. G. Holtzman[2] at the University of Pittsburgh. Holtzman has developed an extensive list of constraints to the timetabling problem in general: other than the four listed above, these are not directly applicable to the Faculty of Business Administration and Commerce at the University of Alberta because they are too specific. However an examination of Holtzman's work is recommended to the developer of

timetabling models because it forms such a broad background.

The University of Alberta generally recommends the scheduling of courses in standard timeblocks of 50 minutes in length on Monday, Wednesday, Friday (MWF) and 1 hour 20 minutes on Tuesday and Thursday (TT).

In order to develop a yearly timetable the Faculty of Business[3] and the Registrar's Office have formulated a list of conditions specifically for this Faculty. (The faculty has approximately 55 full time professors, 1500 undergraduate students and 200 MBA students. In 1971-72 they offered 84 courses in 169 sections and in 1972-73 there are 103 courses offered in approximately 200 sections.) The following is a list of these special conditions.

1) It is preferable that each professor teach a maximum of four days a week.

2) It is desirable but not necessary for classes to start or end on the common-block times and similarly to have them conform to the MWF--TT pattern.

3) A professor should have a ratio of three good times to one poor time. A poor time is defined as 8:00 - 9:00 AM MWF; 3:00 - 4:00 PM MWF; 4:00 - 5:00 PM MWF; 8:00 - 9:30 AM TT; 3:30 - 5:00 PM TT:

4) The set of course sections a professor must teach is given. The number of these varies from instructor to instructor.

5) An attempt is made to schedule single section

required courses between 11:00 a.m. and 2:00 p.m.

All of the above conditions involve members of the faculty or physical resources such as classrooms and none take into consideration the students for whom the timetable is being generated. In some schools,[4] but not this one, a constraint is introduced to ensure that no student has more than a few hours between lectures or that students who live far away from the campus have better timetables because they can not go home to study between classes.

It was decided, however, that with 1500 undergraduates enrolled in the Business Administration program, little attention could be given to student preferences as to times. It is becoming more difficult to accommodate those students who desire a special timetable due to a part time job or other reasons. Therefore, it was necessary for the Faculty to decide that the timetable should ensure that each student is not prevented from taking those courses he wishes-- by conflicts which exist in the timetable. Clearly if a student wishes to take a group of courses for which he is qualified then every effort should be made to make these available to him (no consideration can be given with respect to the convenience to the student of his timetable although this could be built into the timetable criteria).

In order to accommodate the need for availability of courses a "request matrix" is constructed which indicates the number of students requesting each combination of any two courses. In this request matrix a course represented by

a row is compared with all other courses represented by the columns and the elements of the matrix are the number of students enrolled in both the row and the column courses. An example of a request matrix is shown in figure 1. The off diagnal elements of the matrix in fiqure 1 represent the number of students wishing to take both the course represented by the row and the course represented by the column. The diagional elements represent the total enrollment in that course and the matrix is triangular; the top representing the same informaticn as the bottom

	Act. 202	Act. 204	Act. 302	Act. 304	Act. 307	Act. 309	Act. 402	Act. 404	Act. 406	Act. 408	Act. 410	Act. 412	Act. 561	Act. 562	Bus. 201	Bus. 203	Bus. 304	Bus. 306	Bus. 307	Bus. 310
Act. 202	209	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Act. 204	143	442	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Act. 302	4	12	108	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Act. 304	0	5	77	84	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Act. 307	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Act. 309	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Act. 402	0	0	5	4	0	0	48	0	0	0	0	0	0	0	0	0	0	0	0	0
Act. 404	0	0	25	19	0	0	44	72	0	0	0	0	0	0	0	0	0	0	0	0
Act. 406	0	0	10	10	0	0	41	50	52	0	0	0	0	0	0	0	0	0	0	0
Act. 408	1	1	2	4	0	0	10	10	12	17	0	0	0	0	0	0	0	0	0	0
Act. 410	0	0	0	0	0	0	11	12	12	2	12	0	0	0	0	0	0	0	0	0
Act. 412	0	0	7	5	0	0	33	34	35	7	10	38	0	0	0	0	0	0	0	0
Act. 561	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Act. 562	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bus. 201	49	59	2	1	0	0	0	0	0	0	0	0	0	0	430	0	0	0	0	0
Bus. 203	36	46	0	0	0	0	0	0	0	0	0	0	0	0	384	388	0	0	0	0
Bus. 304	26	67	76	62	0	0	1	14	3	4	0	1	0	0	8	4	309	0	0	0
Bus. 306	114	325	12	5	0	0	1	2	1	1	0	1	0	0	43	28	56	390	0	0
Bus. 307	116	336	19	9	0	0	2	4	3	1	1	2	0	0	42	30	78	359	404	0
Bus. 310	108	296	13	7	0	0	0	0	0	0	0	1	0	0	38	23	66	273	279	377
Bus. 312	16	56	72	63	0	0	1	14	1	1	0	0	0	0	4	4	239	31	55	44
Bus. 323	61	294	23	13	0	0	2	4	1	0	0	0	0	0	31	24	75	282	298	258
Bus. 333	65	282	21	11	0	0	1	2	0	0	0	0	0	0	32	25	82	281	297	253
Bus. 357	104	310	19	9	0	0	1	3	1	0	1	0	0	0	32	23	66	307	318	277
Bus. 401	2	9	12	10	0	0	44	52	48	14	13	36	0	0	0	0	19	6	6	12
Bus. 402	0	1	2	1	0	0	4	8	4	1	2	2	0	0	0	0	3	1	1	1
Bus. 406	3	3	8	10	0	0	10	11	9	10	1	6	0	0	0	0	21	2	3	3
Bus. 441	0	1	4	3	0	0	3	5	3	2	0	2	0	0	0	0	10	2	2	0
Bus. 450	0	1	2	1	0	0	0	0	0	0	0	0	0	0	0	0	12	1	1	1
Bus. 452	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bus. 453	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bus. 456	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bus. 458	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bus. 459	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bus. 461	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bus. 462	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bus. 463	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bus. 464	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bus. 470	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bus. 471	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bus. 550	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bus. 552	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bus. 555	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bus. 556	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bus. 566	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bus. 567	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bus. 599	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FIGURE 1
Request Matrix - Faculty of Business
1971 - 1972 Winter Session

The request matrix is, of course, an aggregation of student programs. Without this instrument the problem would be intractable because of the computing time required to test each possible timetable with each student's desired program. It must be recognized that information is lost in the aggregation, but that other, hopefully smaller, problems are created. An example of this latter difficulty is discussed later in the chapter in the context of Multi-Section courses.

The request matrix and the timetable must be constructed long before registration. This leads to the idea of pre-registration--a concept which has yet to be implemented at this or many other institutions. To facilitate the development of a request matrix some form of pre-registration providing a statement of student intentions would be helpful. Pre-registration, however, has some drawbacks such as the difficulty of providing pre-registration counselling, a limited number of returns from existing students and no returns from prospective students. A further drawback is the tendency for students to change their requests for courses between the time of pre-registration and the time of actual registration.

To overcome the foregoing drawbacks the development of the request matrix offers several alternatives: (1) to assume that the returns of pre-registration are a random sample and thus represent a true picture of the course requests, or (2) by the scheduler identifying conflicts

through experience and marking them on the request matrix, or (3) a combination of (1) and (2). It is also possible to examine the effect of different request matrices based on different assumptions about desired student programs.

To aid in the construction of a request matrix for 1972-1973 in the Faculty of Business, a questionnaire was sent out to all students in February 1972. The response to the questionnaire regarding course intent for next year was poor with only 280 out of 1533 students responding and 85 of those responding being 4th year students indicating they would not be returning.

The request matrix in figure 1 gives a clear indication of potential timetable conflicts between single section courses, as well as an indication of possible conflicts in multisection courses. figure 2 is a graphic representation of the conflicts, that has all single section courses offered in 1971-72 listed down the side of the chart. A letter is assigned to each course and is used across the top of the page to represent the same course. The X's in the matrix signifies that at least one student wishes to take both the row course and the column course. Thus an X indicates a potential timetable conflict while a blank signifies that the particular course combination is conflict free. The matrix is triangular because the top half is a mirror image of its bottom.

Figure 2

Conflicts Between Single Section Courses 1971-72

			A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
A	Act.	408																							
B		410	2																						
C		412	7	10																					
D	Bus.	406	10	1	6																				
E		441	2	0	2	5																			
F		450	0	0	0	6	14																		
G	Fin.	401	2	2	4	5	10	11																	
H		430	0	0	1	0	22	11	6																
I	I.R.	401	0	0	1	4	14	8	32	8															
J		411	0	1	4	2	11	7	8	5	16														
K		412	0	0	1	1	2	2	0	1	9	26													
L		414	0	0	0	2	9	3	5	1	4	10	4												
M		421	0	0	1	2	6	5	1	0	0	10	8	3											
N		422	0	0	0	1	4	4	2	2	3	9	6	3	22										
O	M.S.	410	0	0	3	4	0	3	4	1	1	0	0	0	3	2									
P		411	0	0	0	4	0	1	4	1	1	0	0	0	2	2	8								
Q	Mktg.	402	0	0	1	3	0	2	12	3	7	9	5	4	6	2	1	1							
R		403	3	1	4	8	21	17	13	6	21	19	9	13	10	8	1	0	27						
S		405	1	0	1	3	6	2	8	3	7	6	4	1	3	3	4	4	11	10					
T		411	1	0	1	0	5	3	7	1	1	6	5	6	7	5	1	1	9	21	5				
U		414	2	1	1	3	14	4	14	5	9	8	4	6	4	2	5	3	16	30	17	10			
V		418	0	0	1	4	10	5	3	3	17	12	7	8	9	9	1	0	16	36	5	12	23		
W		419	1	0	0	1	5	2	1	1	2	1	1	3	3	2	1	1	4	7	2	7	3	5	

If the problem was simply to avoid putting single section courses which share the same students in the same time block of the timetable the problem would be easier. In the example at hand, there are 30 time periods available (15 first term and 15 second term) and only 23 single section courses to be taught. Thus, assigning one section to each time slot would avoid all conflicts. If there were more than 30 single section courses, then it would no longer be possible to avoid all conflict.

There is, however, another type of conflict which should be avoided. The assignment of a single-section course along with a multi-section course can cause a paucity of certain types of students in the multi-section course. For example, if a single-section accounting course conflicts with one section of a multi-section course, there will be no accountanting majors in the latter section. This is not necessarily of concern but may be undesirable in some situations. The introduction of multi-section courses into the system causes the problem to become increasingly complex.

With multiple section courses it becomes impossible to trace the conflicts which might occur in a particular student's program because this information has been aggregated into the request matrix. Certain courses may have several sections each, yet it may not be clear even after an examination of the timetable, whether conflicts actually do exist for individual students. It is generally not

necessary, to assure individual program feasibility, to offer all sections of a multi-section course at different times from those of another multi-section course, even if the request matrix shows that a number of students want to take both courses. In fact, it is usually impossible to develop a timetable which avoids all conflicts for all multi-section courses. On the other hand, using individual student programs to test the timetable is not feasible because of the computer time involved; so on practical grounds the request matrix route seems the best way to structure the problem.

Reasons for Undertaking this Research

There has been no lack of theoretical research done on class-teacher-timetabling problems, but our investigation has disclosed that there has been a dirth of practical usable research on this problem. The problem itself is an important one because every year each department of every academic institution must create a timetable. Each year a workable timetable is generated, usually by hand for each institution, yet no widely acceptable computerized method has been uncovered. Many systems have been developed to construct a timetable but either because the research over simplified the real problem, or because there is a lack of computing power available, or both, these methods are not usable.

It was the lack of computer success with practical timetabling tools and the characteristics of the class-

teacher timetabling problem, as well as other kinds of timetabling problems, which led to the focus on university timetabling.

Aim of the Research

As stated earlier the purpose of this research was to develop a computer program which would aid the human scheduler in the creation of a workable timetable in a shorter period of time than previously required. In order to achieve this end the FRED (For the Reduction of Education Dirtywork) system was developed. It is a heuristic computer program which aids the scheduler in the preparation of an acceptable timetable in a reduced amount of time. The system allows the scheduler to sit at a terminal and interact with the computer in developing and improving his timetable. He is able to modify an existing timetable and check for conflicts and continue modifying the timetable until he is satisfied that it is one which will provide a working solution for the coming year. Hopefully with FRED this timetabling development process will take only a part of a day rather than the several months of part-time activity it presently requires.

The system was designed in such a way as to take into consideration two computer aspects . First, the computer at the University of Alberta is an IBM 360/67 under MTS (Michigan Terminal System) which makes terminal use economical and advantageous . Therefore, it was decided to

produce a system which would be developed totally by using on-line facilities such as the IBM 2741 typewriter terminal or CRT display screens.

Secondly, PL/1 is an excellent language for handling character data programming problems with its use of structures and storage allocation. It was not clear how efficient or convenient PL/1 would be used solely as a terminal language. Therefore some examination of its effectiveness was required and is described more fully in chapter four.

References

1. Chapman, Randal G. "Dynamic Programming of the School Timetable". (Unpublished Research, University of Alberta 1971).

2. Holtzman, A. G. "Optimal Scheduling in Educational Institutions", University of Pittsburgh, 1964.

3. Discussion with D. A. McGeachy, personal interview, July 21, 1972.

4. Swedish school for which Greko did his research.

Chapter 2

The Present Timetabling Procedure for the Faculty of Business at the University of Alberta

This chapter is devoted to a description of the timetabling procedures used by D.A. McGeachy, assistant to the Dean of the Faculty. These procedures are not necessarily those used elsewhere, ncr are they necessarily the best non-computer methods available.

Preparation of the Initial Timetable

In the Faculty of Business Administration there are five Area Groups (Economics, Organzational Behavior, Quantitative Methods, Accounting and Finance); each Area Group has as its chairman an Area Co-ordinator. These Area Groups are responsible for all matters pertaining to that subject area including the assignment of professors to courses and the determination of the number of sections of each course.

From each Area Co-ordinator the Scheduler receives a list of courses that are to be taught by that Area-Group in the coming year. Along with the name of the course, the number of sections of each course is supplied. In order to obtain the number of sections an estimate of course enrollment is made and divided by the desired class size. This estimate is derived from past history and any indicators the Area Co-ordinators feel they can rely upon -

however it should be made clear that no formal attempt is made to obtain information from the students.

Starting with a blank timetable all multi-section courses are assigned to time periods with an attempt to maintain the ratio of 1 in the morning to 1 in the afternoon, randomly distributing the courses throughout the week. For example a course may have 2 sections. One might be scheduled MWF 10:00-10:50 and the other TT 2:00-3:30. When all multi-section courses have been assigned the single section compulsory courses are placed in the middle block of the day (i.e. 11:00-2:00) and all single section non-compulsory courses are assigned where they fit into the instructors' schedules and so that they do not appear to cause student conflicts, subject to the intuition of the Scheduler. With regard to the number of concurrent courses a balance between time blocks is sought as well as a balance between the number of courses each term. If the Faculty of Business fails to assign enough 8:00 o'clock classes the Registrar will modify the timetable because there are not enough rooms to hold all classes at good times and the poor times should be distributed equally amoung faculties.

Out of every 4 courses that are scheduled one should be given a "poor time" as defined in Chapter 1. By following the rule of a 1 to 3 ratio, the timetable has a more even distribution of courses per time period.

The timetable is produced long before faculty recruiting is complete. Thus the timetable is designed on

the assumption that the recruiting will be successful. If at a later date the Faculty discovers that some prospects will not be joining the staff then changes will have to be made to some courses. The preparation of the timetable to this point takes approximately 1 1/2 man-days.

Next the timetable is checked by hand in an attempt to search for any discrepancies that will prevent students from taking the desired pattern they have developed. This checking is done by consulting a conflict matrix prepared by hand which shows all the courses being offered in one time period. The Scheduler then matches each course with each other course and determines by intuition whether or not these courses should be offered concurrently. Due to the fact that no student requests have been obtained, this conflict checking can only be a gross approximation based on past experience, and estimates about the coming year. Also a human checker is never perfect and a conflict could easily go undetected.

The timetable is then sent to the Registrar of the University, who inspects it for conformity to the University's Administrative rules of timetabling which comprise some of the constraints in Chapter 1, and assigns rooms to each course section. This is done on a first come first served basis so if the faculty wishes to obtain rooms in a certain building it is essential to submit the timetable early. The timetable is then returned to the Faculty.

Following this, the Area Co-ordinators submit lists of courses that each professor in their Area Group is to teach. No times or sections for each professor are indicated at this point. McGeachy then assigns professors to the sections in the prepared timetable. It is desired (although unofficially) that a professor teach no more than 4 days a week and that each professor must take one undesirable time out of four. Each professor has certain preferences for teaching times (or in some cases times that he is available to teach) and each tries to influence the Scheduler in order to obtain his desired timetable.

Therefore after the assignment of professors to times and sections the timetable is complete. It is typed and distributed to the faculty members for comment. It is common at present to continue to make changes to the timetable for several months (one year 15 revisions were made) and well into September and registration week. Reasons for these changes vary from the professor's desire or preference for times; his lack of availability; room size (too many students) or room acoustics. It is rarely changed because of student conflicts since these were avoided at a prior step in the process.

The Advantages of the Present System

There are two main advantages to developing the timetable using the present system.

The first advantage is the flexibility that the human Scheduler brings to the development of the timetable and in dealing with modifications of the timetable. The human Scheduler is aware of the preferences of the staff; all the constraints imposed by Students, Faculty and Administration and he is able to make value judgments .

Secondly the system is relatively inexpensive, costing only 1 and 1/2 man-days at $75.00/day (approx.) to create an initial timetable and several man-days of time spread over many months, to make the modifications requested.

The Disadvantages of the Present System

Under the present system it is not necessary to clearly define the objective of timetabling or state clearly the problem. Only when defining the bounds of the problem for computerization is a formal specification required. This causes the objectives not to be totally met nor the constraints to be totally binding. Thus one disadvantage of the present system is that no attempt has been made to produce a formal set of objectives, constraints and parameters required in developing a yearly timetable.

Secondly, once the timetable has been formalized the checking procedure to determine conflicts for both students and faculty is poorly defined and allows errors to pass. This is not a criticism of the Scheduler, for the task of ensuring all conflicts have been dealt with is time

consuming and inevitably some conflicts pass without discovery.

Thirdly, very little of the present input to the system is concerned with the students. Most input concerns faculty and their preferences, and administrative regulations set down by the Registrar's Office. Furthermore, the request matrix is only a "guess" as little information regarding students has been obtained or included. Following from this is another weakness. The number of sections for each course is determined by "estimated enrollment." If the number of sections is incorrect a large amount of shuffling is required to remedy the situation. Too many sections and insuffiecent students leads to small sections or a section with no students. The converse leads to overcrowded rooms and poor teaching conditions. However errors in enrollment estimates are common. Until better information can be gathered and used this guessing will lead to problems in sectioning of courses.

In conclusion the weaknesses of the present system are lack of clearly specified objectives, constraints, parameters and scheduling procedures, poor checking procedures, little consideration of student desires and poor estimates of future enrollment.

Chapter 3

The State of the Art.

The Algorithmic Approach

One approach to solving the class-teacher timetabling problem is to use a mathematical algorithm and a computer to generate a solution. Among existing algorithms one finds many management science/operations research techniques such as linear programming, dynamic programming, flows in networks and critical path scheduling.

The appeal of the algorithmic approach lies in the desire to create an optimal timetable - which is what these O.R. tools are designed to do. However, on closer examination it is hard to decide how to measure the optimality of a timetable. One could optimize from the student, professor or administration point of view and within each catagory there are many obvious objectives to be optimized. A combination of the three points of view could be used to optimize the timetable with weighted objectives comprising the overall criteria.

The first major work that was done on an algorithmic approach was by C.C. Gotlieb[1] in 1963. Gotlieb's work matches a requirements matrix against an availability vector and through successive iterations of his algorithm a feasible solution is found. This method is useful for an educational system where one set of students belongs to the

same class irrespective of what subject is being taught. This system results in a relatively simple scheduling problem because of the elimination of many different individual programs. The university environment, however, makes necessary the use of an individualized program for each student according to his desires and needs.

Gotlieb's work led to further study of the algorithmic approach and in 1965 a Swedish operations researcher named Bert Greko[2] formulated the timetabling problem expressed as a flow through a network. Greko chose a model because from his experience in industrial planning he felt that this tool was not only useful for producing feasible and optimal schedules but also helps clarify complex problems to non-mathematicians. Greko regarded the total system of the school scheduling problem to be combinatorially large and unstructured.

A network is a mathematical respresentation useful in allocating scarce resources to various competing activities in maximizing benefit or minimizing cost. The network consists of a series of points called nodes which represent the various activities and a series of arcs which "join" the appropriate activities. These arcs control the allocation of resources to the activities by specifing a maximum and minimum quantity that can be allocated to each node and a cost associated with the use of that resource for that activity. For a more detailed discussion of networks see Oystein Ore[3] "Graphs and Their Uses".

In Greko's model there are five types of nodes. Type one are "elementary groups" which is Greko's term for unique combinations of courses that a student can take. The second type of node represents the teachers - one for each teacher. The third type is the periods that the teacher could teach in and the fourth set of nodes is the courses taught in each period. The fifth set represents the courses taught by the school - one node for each course. For a picture of Greko's network see figure 3. Greko's model was run for a high school of 188 students and 35 teachers. Each student must select 1 course from each of 4 groups of courses. This substantially reduces the number of combinations available, rather than the university situation where a student may choose 10 or more courses from all those available. Greko's model can be used to describe the university situation but the size of the problem is enormous and the cost of obtaining a solution prohibitive. It is not unreasonable to believe that the Faculty of Business has 500 unique patterns (a pattern represents a group of courses which a student is taking) each of which has 10 courses on the average. Therefore the number of nodes and arcs required to formulate a Greko network is horrendous and the man-hours required to formulate the data and the computing time required to complete such a task makes the implementation of this method infeasible.

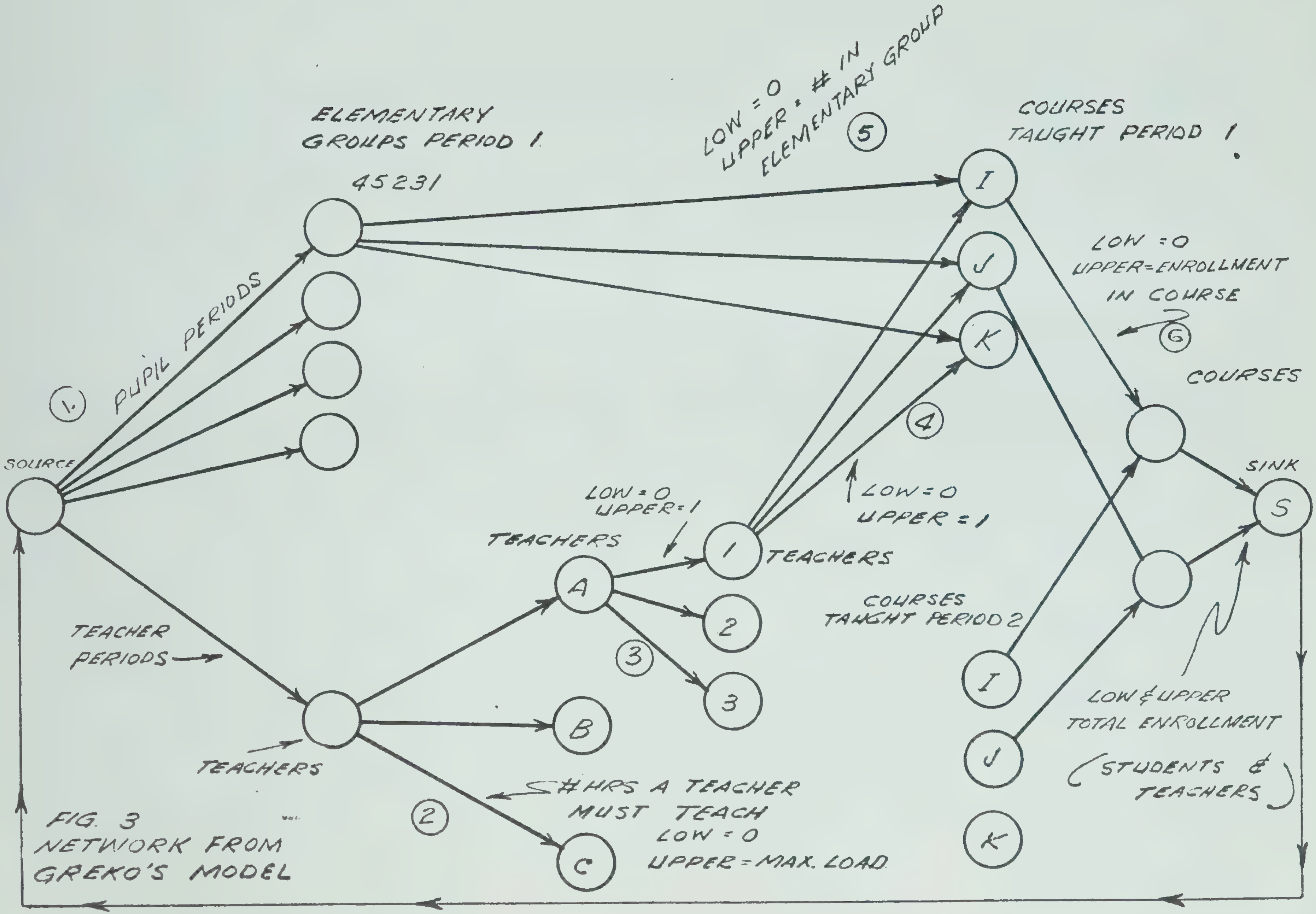

FIG. 3
NETWORK FROM
GREKO'S MODEL

In 1970 Dominique de Werra[3] of the University of Waterloo described the timetabling problem as a bi-partite multi-graph which is partitioned into sets. The simplest form he describes consists of two sets: C (classes) and M (teachers), where class C must be assigned to only one teacher in the set of M. de Werra says this procedure has been used to construct timetables for Swiss schools with about 50 classes and 80 teachers for 35 periods a week.

Combining the work of de Werra, and Maher[4567] a new network model was designed to reduce the amount of data required; this model applies to the non-fixed class category of timetabling problem. An Out-of-Kilter[8] algorithm was formulated (see figure 4) which represents constraints as described in chapter 1. An out-of-Kilter algorithm is one in which the sink and source are joined and the flow into the sink equals the flow out of the sink to the source. Thus no node retains any resource.

Starting from the source the first set of nodes represents courses which are to be taught, one node for each course (i.e. Accounting 202, Accounting 204, etc). The arcs connecting the source and these course nodes represent the maximum number of sections of each course. For example if it is desired to teach a maximum of 8 sections of Accounting 202 then that would be the arc capacity. Usually 8 would also be the minimum because in practice the number of sections of a course is fixed for the purposes of timetabling.

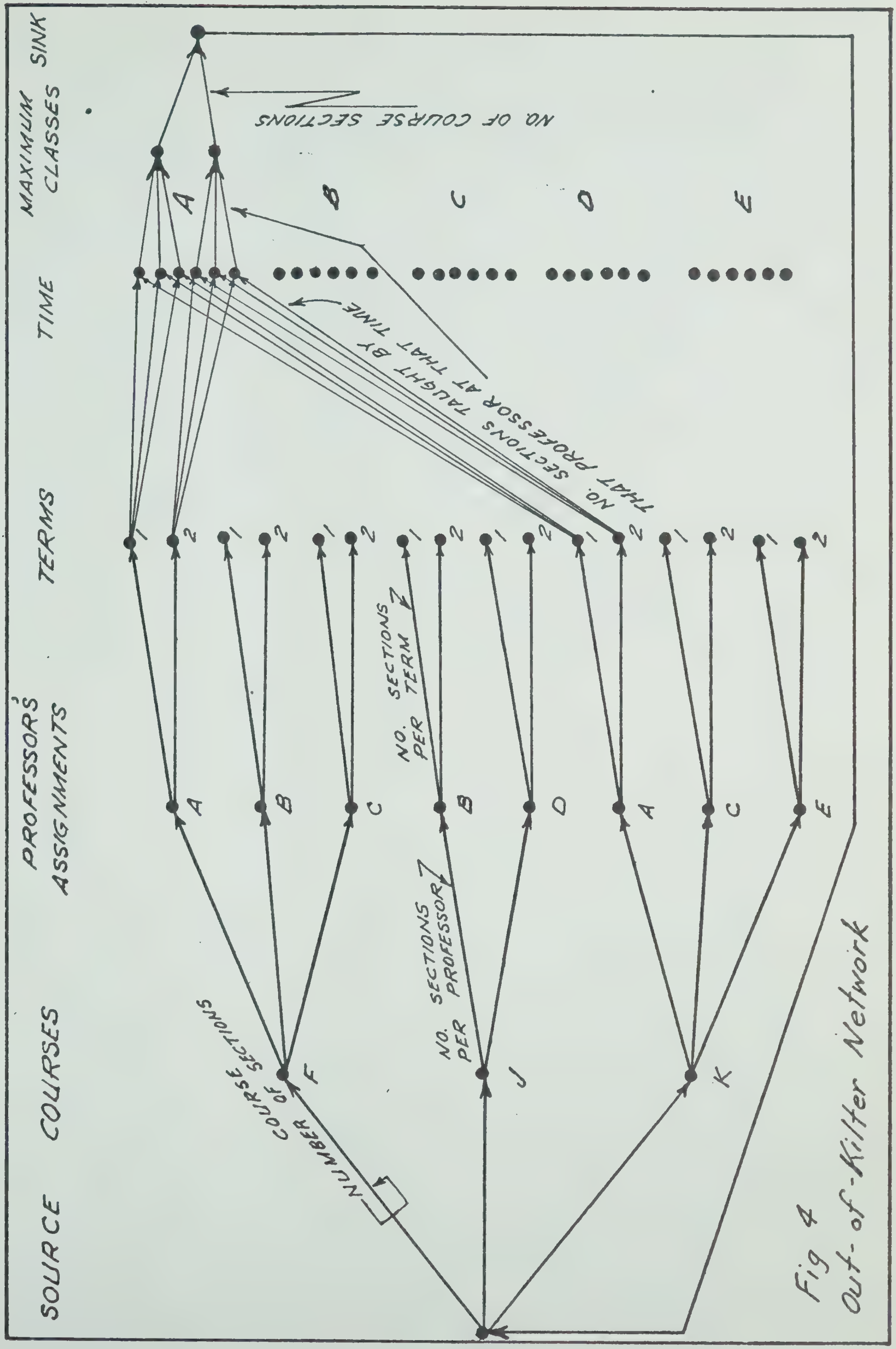

Fig 4
Out-of-Kilter Network

However this need not be the case and the minimim and maximum arc capacities could be different thus leaving the decision on the number of course sections to offer to the algorithm.

The second set of nodes represents professors. The arcs leading from the course nodes connect with each professor who could teach that course. The capacity of the arcs are the minimum and maximum number of sections which each professor is required to teach during the coming year.

The third set of nodes in the network is marked terms, and represents the terms which these course sections could be offered. Each professor might teach a section in each of the two terms and therefore the arcs go from the professor to the term nodes. Because there are 2 terms there are twice as many term nodes as professor assignment nodes at a maximum. The capacity of the arcs connecting these nodes represents the minimum and maximum number of course sections each professor could teach in that term.

The fourth set of nodes (time periods during the work) is the key set and has a weighted value function associated with it. It is the sum of weighted flows through these nodes which is to be maximized. Each professor has 30 nodes, one for each time period in each term. All the course sections assigned in the network to a certain professor are connected by arcs to the professors' time blocks, those first term courses to the first fifteen time periods, those of the second term to the last fifteen. The capacity of the arc is a maximum cf 1 because of the constraint which says that a professor can only teach once during any time period. The mimimum capacity is 0 which indicates that the course section in question need not taught in any particular time

periods.

The fifth set of nodes represents the maximum number of sections which can concurrently be taught in any time period. There are 30 of these nodes and all course sections taught in time period 1 first term are connected to node 1. The maximum capacity is equal to the maximum number of courses desired. The minimum capacity represents the fewest number of courses to be taught concurrently.

The arcs leave the maximum class node and flow to the sink. The sink has an arc connecting it to the source thus creating an Out-of-Kilter network. In order to establish this network it is essential to know how many sections are offered. This Out-of-Kilter network reduces the data requirements for a computer run well belcw that required for the Greko model, but it still requires a sizeable amount of effort to get the data into a suitable form.

The Out-of-Kilter model can be made to account for the tctal load for each professor if one is willing to assign professors exactly to course sections and thus give up some generality. However, we have not been able to insert the notion of avoiding student program conflicts into the model even theoretically, and if we could, the problem size would be overwhelming.

An additional drawback in the solution of the timetabling problem by network flows is the necessity for integer solutions. Since Out-of-Kilter networks are a subset

of linear programming problems there is no guarantee that the solution will be integer. In the timetabling problem a non-integer solution is meaningless.

In order to obtain an integer solution Swart[9] developed an integer programming model for imbedded unimodular constraint matrices. Swart defines his model:

> "... the constraint set of the course scheduling problem can be partitioned into two sets, one possessing the unimodular property. Furthermore it will be shown that there are, in general, three different manners in which the constraints may be so partitioned. These will be denoted by the SIP, the MIP, and the LIP, partitioning schemes respectively."

Swart runs all 3 schemes on a small medium and large timetabling problem. The results of his efforts are presented in figure 5,6 and 7.

It can be seen from figure 7 that his largest problem contains 400 variables. A variable in this context is X = 1 if course j is taught in line period i; 0 otherwise. This would imply 40 courses offered over 10 time periods. At the University of Alberta Faculty of Business there are 103 courses offered in the 1972-73 winter session and 30 possible time blocks. This gives 3090 variables and causes the problem to be unmanagable. If by "course" Swart means "section" as defined by this University then the number of course-sections would be closer to 200 and would yield approximately 6,000 variables .

Not only does the problem size under Swart's formulation present computing difficulties, but there is a

theoretical problem as well: Swart must indicate as part of the constraint set which courses (or course-sections) must not be offered at the same time, but in the case of multi-section courses it is not possible to do this, and so the method fails.

We have presented examples of the best algorithmic models available. The only one which accurately captures the problem is Greko, and as noted above, both the amount of data and the computing time required, and the cost, make this approach impractical.

Figure 5. Summary Presentation of Problem One with Solution Times

Partitioning Scheme	No. of Variables	No. of Nodes	No. of Arcs	Number of Additional Constraints	Process Time	Input-Output Time	Total Time
SIP	30	14	43	35	0.547 min	0.253 min	0.80 min
MIP	30	18	62	30	4.18 min	0.106 min	4.28 min
LIP	30	24	52	25	0.460 min	0.241 min	0.70 min

Original integer programming formulation: 30 variables
46 constraints.

Figure 6. Summary Presentation of Problem Two with Solution Times

Partitioning Scheme	No. of Variables	No. of Nodes	No. of Arcs	Number of Additional Constraints	Process Time	Input-Output Time	Total Time
SIP	115	30	144	105	26.85 min	0.253 min	27.2 min
MIP	115	45	374	90	8.46 min	0.304 min	8.75 min
LIP	115	85	199	50	Optimality achieved, not verified > 60 min.		

Original integer programming formulation: 115 variables
135 constraints.

Source: Swart, Ibid.

Figure 7. Summary Presentation of Problem Three with Solution Times

Partitioning Scheme	No. of Variables	No. of Nodes	No. of Arcs	Number of Additional Constraints	Process Time	Input-Output Time	Total Time
SIP	400	52	451	290	No optimality achieved > 60 min		
MIP	400	62	651	280	No optimality achieved > 60 min		
LIP	400	192	591	150	Optimality achieved--not verified > 60 min.		

Original integer programming formulation: 400 variables
340 constraints.

Source: Swart, Ibid.

Heuristic Models

The other approach to solving the timetabling problem is by use of heuristic methods. The key work done in this area was initiated by Mary Almond at the University of London.

The Almond method considers the problem as a three dimensional matrix which represents a blank timetable (see figure 8) and proceeds by selecting entries from the requirement matrix one-by-one and allocates it to a suitable time and lecturer to satisfy the complex conditions of timetabling. If any allocation fails some conditions are changed--the timetable is wiped clean and the whole process started all over again. This method was used to develop a timetable for the Mathematics Department and Science Faculty of the Queen Mary's College at the University of London. Individual student requests are not considered at all. The computer program was written in ALGOL.

In September 1971 Edward F. Smith[10] built a four step computer package that

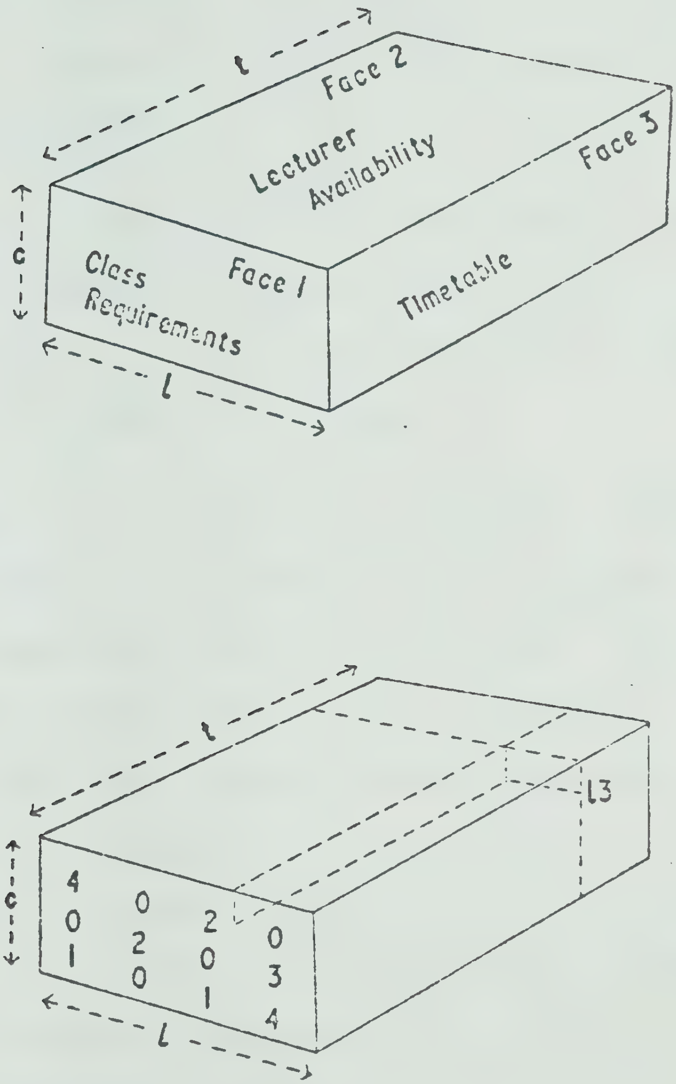

Figure 8
Figurative Representation of Almond Algorithm

implemented the Almond Algorithm for St. Laurence college in Ontario. According to Dr. Smith the program functions adequately for a small college, but the program has not been tested under the differing conditions of the university environment. The computer program runs on batch processing and is written in FORTRAN.

As reported by George Newfeld[11], Elizabeth Barraclough also developed a heuristic program for creating a timetable which was designed for a small grade-school; it would not be appropriate for university timetables.

In conclusion the underlying idea in developing a heuristic approach to timetabling is to create a master schedule which eliminates conflicts for the faculty and reduces to a minimum the conflicts for students. In order to achieve this goal the methods assign courses to a blank timetable until infeasibility is reached, then the courses are re-arranged. It appears that less effort has been spent on research on improving the heuristic method than has been spent on the algorithmic approach.

<u>Algorithmic vs Heuristic</u>

The advantage of an algorithmic approach is the development of an optimal timetable. The use of electronic computers makes available tremendous amounts of computing which unfortunately do not seem sufficient to solve the problems which are faced by academic schedulers.

The Heuristic approach does not attempt to find an

optimal solution but rather a feasible and satisfactory one. To date, though, most heuristic methods have involved some method of trial and error which can prove very costly with computer time and may or may not generate a feasible solution.

Timetabling work done by A.G. Holtzman

Perhaps the most comprehensive work on the timetabling problem was done by A.G. Holtzman[12] at the University of Pittsburgh.

Holtzman examines a wide range of aspects of timetabling from optimization to heuristic methods. His objective is to develop a new tool for timetabling without considering any of the present systems. The work done is comprehensive and well documented, but by his own admission more research is required before actual implementation. In spite of this,

ReferencesProc. IFIP Congress 692 Munich, North Holland Publishing Co. Amsterdam, 1963.

2. Greko, B. School Scheduling through Capacitated Network Flow Analysis; The Swedish Office of Organization and Management, 1965.

3. Orc, Oystien. Graphs and Their Uses The L. W. Siager Company 1963.

4. Norman R. Baker, C. Richard Shumery, P. Micheal Maher and William E. Souder. An R & D Project Selection and Budget Allocation Model for Large Decentralized Research Organization.

5. Maher, P. Michael Documentation for Optyag: A Research and Development Resource Allocation Program (Working Paper, Faculty of Business Administration and Commerce University of Alberta, 1971).

6. Maher, P. Micheal Operating Manual for WES841 A Resource Allocation Computer Model (Working Paper, Faculty of Business Administration and Commerce University of Alberta, 1971).

7. Baker, N.R, Shumery C. R. Maher, P.M; Souder

W.E. and Large R.D. *AMC Value Methodology Experiments; Session at AIRLIE House: Material Presented to Attendus.* Faculty of Business Administration and Commerce University of Alberta, September 1970).

8. Network was developed by the author with the help of N. R. Baker at the Georgia Institute of Technology and W. E. Souder of North Western University.

9. Suarts W. W. *Integer Programming for Imbedded Unimodular Constraint Matrices with Application to a Course-scheduling Problem* (Unpublished PL.D. disertation, Georgia Institute of Technology, July, 1970).

10. Smith C. F. "Computer Assisted Timetabling on a small college computer". (Canadian Datasytems September 1971 p37).

11. Newfeld G. A. "Academic Scheduling Project Report" Research report for Administrative Data Processing, University of Alberta, October, 1970).

12. Holtzman, A. G. Ibid

Chapter 4

A Man-Machine Interactive System in a Heuristic Program for University Timetabling: The FRED System.

In most timetabling research conducted up to now computer programs written to utilize a heuristic model have required the formal specification of the heuristics desired. This specification is necessary to be able to formulate a computer program which will produce a final workable timetable without requiring further human modification. As has been noted in Chapter 2 it is not always possible to express verbally an intuitive process such as how to rearrange courses when a conflict arises. In cases where formal specification of the heuristic is not possible programming the timetable is not possible either.

As is pointed out in Chapter 3, the present heuristic method of developing a timetable is to assign courses one-by-one until a constraint prohibits the assignment of another course. At this point the inputs to the program are reshuffled; the slate or timetable is wiped clean and the one-by-one assignment is repeated in a different order. The differences between the several heuristic programs which are available is the method of reshuffling the input data. It is clear that this type of heuristic is not a true description of the way a person develops a timetable. If the human timetabler was to perform a heuristic he would not follow such a "sledge hammer" approach. Therefore even when

heuristics are specified it appears that so far they have not been adequately duplicated in a computer program.

The idea for a terminal interactive system was derived from A.C. Brewer[1] who developed an interactive scheduling system to allocate tasks for the Manned Space Flight Center, NASA. Brewer's system (Computer Assisted Network Scheduler, CANS) supports both manned and unmanned missions for NASA up to one year without conflict. The interactive system provides the ability to modify and observe actual or simulated changes. It enables NASA to ask the question "what if". The theory behind the present research is that if it is costly to include heuristics in a computer program, and when this is done it is not a true description of the real world, then the heuristics should be left to the man. This goal can only be achieved by using an interactive on-line computer system so that the man interacts with the computer as the program executes.

The Man's Role in the FRED Scheduler

In the FRED scheduling system the human Scheduler is asked questions by the computer program. Due to the facts previously discussed, it is easier for the man to develop a heuristic timetable than for the machine and so that is his function. The molding, shaping and development of the timetable is left entirely up to the Scheduler. He makes all decisions. Thus he has the ability to modify, change, rearrange or even change his mind about the approach to the problem without totally destroying all work and thought put

into the system.

Because of the complex set of constraints which the timetabler understands he is able to make quick evaluations as to the acceptablity of a certain timetable. For reasons which might be obscure, involved and hard to program the human Scheduler can accept or reject a timetable the computer presents to him.

<u>The Computer's Role</u>

The man in the man-machine system performs all the tasks associated with most of the clerical tasks and the computer's task in the FRED system is to arrange the data and display to the human the pertinent information at that instant. It allows the Scheduler to perform any desired task with regard to the timetable: FRED does <u>not</u> make any decision with respect to what is best. The computer does give an evaluation of the proposed schedule, in terms of number of student conflicts or potential conflicts, and thus gives the sheduler some notion of the progress he is making. No evaluation of professor preference or administrative preference is made, however, the scheduling of a professor for 2 classes at the same time is avoided. Therefore, any timetable derived by the system is feasible for all professors. In addition, data organization, manipulation and maintenance is handled by the computer. Thus we have obtained the best effort from both parts of the System.

<u>Input to the FRED System</u>

As with the hand generated method much information is required about faculty desires. A list of all courses offered (courses can be offered but not taught) by the faculty are inputed along with a numeric code for each course. For example ACT202 is a course and its code is 001 while ACT204 is another course whose code is 002. Secondly a list of all Professors and their numeric codes is inserted. Professor ABRAHAM is 01. Professor names are only 1 word, usually the last name; in the case of two faculty members with the surname of Jones (for example:Peter Jones and Ken Jones) the names are written as PJONES and KJONES or PETER-JONES and KEN-JONES, etc. It is essential that on every successive mention of that faculty member that it be referenced in the same manner.

It is common for the timetable to be made up long before faculty recruiting is complete and therefore some professors are tentative or unknown. The FRED system is capable of handling such occurrences. For example, if 15 prospective faculty members are to be scheduled into the timetable their names could be used or such pseudo names as NEWPROF1, NEWPROF2 can be used. Any number of professors can be inserted.

The third input is a version of a timetable. This need not be complete but may have as many course sections as have been determined. A null timetable is also acceptable thus allowing the Scheduler to start from scratch. The information required on input is:

a) The six digit name, e.g. ACT202

b) The section number, e.g. B1

c) The Term, e.g. 1

d) The Starting Time, e.g. 8:00

e) The Ending Time, e.g. 1420

f) The Day(s), e.g. MWF

g) The Professor, e.g. Winters

h) Estimated Enrollment, (not necessary) e.g. 23

I) Sequence Number, e.g. 30

The Sequence number is a number assigned to a course by the Registrar, according to time period and number of sections. A single section course is given the sequence number 10 or 11 and multi-section courses have numbers of 20 or greater. The sequence number is used to determine whether a section is a single section course or one section of a multi-section course.

The FRED system then loads the timetable into memory using the 16 standard time blocks (see table 1). No matter when the course starts or when it ends the program will assign it to one or more of the standard time blocks. If a course takes only a portion of a time block, for conflic-checking purposes it appears that the entire time block is being used. Thus, two successive courses scheduled in a non-standard manner might spuriously appear to be in conflict, e.g., Course A from 14:30 to 15:30 and Course B from 15:30 to 16:30 both occupy the standard time block 15:00 to 16:00. Such a problem arises rarely. It would have been possible to

divide the week into shorter periods (half-hour for example) to avoid this problem, but prevailing University custom uses the standard time blocks, and in similarly using them the data and programming requirements for FRED were greatly eased.

Table 1

Standard Teaching Time Blocks at the University of Alberta

Number	Days	Hours
1	MWF	8:00 - 8:50.
2	MWF	9:00 - 9:50.
3	MWF	10:00 - 10:50.
4	MWF	11:00 - 11:50.
5	MWF	12:00 - 12:50.
6	MWF	13:00 - 13:50.
7	MWF	14:00 - 14:50.
8	MWF	15:00 - 15:50.
9	MWF	16:00 - 16:50.
10	TT	8:00 - 9:20.
11	TT	9:30 - 10:50.
12	TT	11:00 - 12:20.
13	TT	12:30 - 13:50.
14	TT	14:00 - 15:20.
15	TT	15:30 - 16:50.
16	any day	any times other than 8:00 to 15:50

The fourth input is the "Request--Matrix." Here the FRED System differs from the present manual system in that information about students is used directly. This matrix gives the number of students wishing to take both Course A (row) and Course B (column) of the request-matrix. No attention is given to sectioning. The diagonal of the matrix is the total enrollment in that course.

In order to obtain information for 1972 - 73 a pre-registration questionnaire was sent out but very poor response was obtained (see figure 9).

If pre-registration is impossible or quite incomplete then the human Scheduler must make estimates of courses in conflict. He can do this in two ways. Either he can make estimates of the actual number or he can use 1 to indicate a potential conflict and 0 no conflict. The other alternative is to use last years' request matrix.

FIGURE 9

PRE-REGISTRATION QUESTIONNAIRE

UNIVERSITY OF ALBERTA
FACULTY OF BUSINESS ADMINISTRATION AND COMMERCE

TO: All Students. DATE: February 10th, 1972.

FROM: Christian T.L. Janssen,
Associate Dean.

PREREGISTRATION QUESTIONNAIRE

In order to complete the timetable we would like to know the courses you plan to take next year (required and optional courses, as well as any you intend to repeat).

Most courses to be offered next year can be found in the current calendar. (Next year's calendar is not yet available).

We will regard your answer as an expression of interest and you will not be held to it should your intentions change over the summer.

WOULD YOU PLEASE COMPLETE THE QUESTIONNAIRE BELOW

1. HAY
(Surname)

2. 709379 I.D. #

3. G. BRUCE
(Given Names)

4. Current Year of Program.
(Please Circle). 1 2 3 4

PLEASE CIRCLE THE COURSES YOU WISH TO TAKE

Course		Code
Acc.	202	01
"	204	02
"	302	03
"	304	04
"	402	07
"	404	08
"	406	09
"	408	10
"	410	11
"	412	12
Bus.	201	16
"	203	17
"	306	18
"	307	19
"	310	20
"	312	21
"	323	22
"	333	23
"	357	24

Course		Code
Bus.	365	25
"	401	26
"	402	27
"	406	28
"	409	29
"	415	30
"	419	31
"	441	32
"	442	33
"	450	34
Fin.	304	59
"	401	60
"	411	61
"	421	62
"	430	63
"	432	64
Ind.R.	304	69
"	401	70
"	411	71
"	412	72
"	414	73

Course		Code
Manag.	356	76
"	357	77
"	410	78
"	411	79
Mark.	304	85
"	402	86
"	403	87
"	405	88
"	407	89
"	411	90
"	414	91
"	418	92
"	419	93
Org.T.	303	96
"	305	97
"	421	98
"	422	99
"	430	100
"	440	101
Prod.	431	102
"	432	103

I will not be returning in the 1972/73 session:
a) I will graduate this year ____
b) for other reasons ____

PLEASE PLACE YOUR COMPLETED QUESTIONNAIRE IN
THE BOX PROVIDED.

Thank you.

The Functions of the FRED System

In order to provide an aid to the human Scheduler the FRED system has been designed to perform the following functions:

I. LIST THE TIMETABLE

The system is able to produce a timetable which is printed in a format that is acceptable for distribution. The entire timetable can be produced or listed by course or by time block.

II. CHECK FOR CONFLICTS

The routine checks all time blocks and inspects for conflicts between any two single section courses and refers to this as a direct conflict. If there are any students wishing to be in both sections then a message is printed on the terminal informing the Scheduler that such an occurrence has happened. The other type of conflict is called "indirect" and occurs when a single section course and a multi-section course are offered at the same time. If there are any students wishing to take both courses a message is again generated. Note that no other action is taken by FRED:it simply informs the Scheduler that such conflicts do exist.

III. LIST THE PROFESSORS' TIMETABLE

The FRED system will allow for the production of timetables by professor. Either a complete list of all

professors can be produced or only timetables for specific professors at the request of the human Scheduler. These timetables are printed in two columns, one for the first term and one for the second term.

IV. DELETE A COURSE SECTION

This routine allows the deletion of a course section from the timetable.

V. CHANGE A COURSE SECTION

For any course section the Time, Day or Term can be changed and the course will then be placed in the appropriate timeblock and removed from it's present location. Thus the entire timetable can be shuffled around by using the change routine and a completely new timetable generated. This allows the Scheduler to ask "What If?" and see what effect certain combinations of classes will have.

VI. REASSIGN A PROFESSOR

The only item that was not changed in the CHANGE procedure was the professor who was teaching that course section. This routine is able to perform that function. It allcws courses to be taken away from professors and assigned to others in order to build a timetable which provides even teaching load distribution.

VII. INSERT A COURSE

This routine allows the insertion of a new course or section which is not already included in the timetable. However, this course must already exist in the list of

courses provided for in the request matrix. The same information is required here as was required when initially reading the timetable to start the program.

VIII. SAVE THE TIMETABLE

After modifying the timetable for a while the human Scheduler might not wish to continue but would like to retain the work he has already done. This is provided for by the SAVE routine which writes out the modified timetable as well as stores it in file to be retrieved at the next session. He can also have intermediate solutions in case modifications of them do not prove fruitful.

IX. MODIFY THE REQUEST MATRIX

The FRED System also provides the ability to modify the request matrix should some new information become available to the Scheduler. This routine allows the changing of the value of the elements in the request matrix which would permit sensitivity analysis in the scheduling problem. It is important to note that these changes are only temporary and are not maintained once the user has signed off the system. Thus these changes must be made at each successive sign-on, but such temporary modifications are possible (a feature to save the modified request-matrix could easily be built).

X. STOP

The final routine allows the user to sign off in the normal manner.

Thus the ten routines allow the human Scheduler

complete flexibility in preparing the timetable. By selecting the routines he wishes he can modify the timetable until he has a version which he deems suitable, at which time the total timetable can be saved and printed.

PL/1 As a Source language

In programming the FRED System a source computer language had to be chosen. Because the problem consists primarily of character manipulation FORTRAN was ruled out as being inefficient. COBOL, on the other hand, is not well supported at the University of Alberta and is suspect as a terminal Language. Also it does not possess substantial array features. PL/1 possesses the best features of both languages although some investigation was required to determine the appropriateness of PL/1 to develop a terminal system.

The advantages of PL/1 are its handling of character strings and structures facilitating the storing of the timetable. The array of structures feature was used heavily. The disadvantage found was the expense of compilation and the length of time required to compile at a terminal. One important feature we added was a routine to screen terminal input for errors so that such errors do not cause the program to abort. A sample terminal session is shown in Appendix I.

References

1. Brewer A. C. "Interactive Scheduling System" IBM Systems Journal Volume 10 number 1 1971).

Chapter 5

Results of Application of the FRED System

This chapter is devoted to an examination of the applicability of the FRED system to large scale timetabling problems in actual practice. To accomplish this objective the procedure followed in the creation of a timetable are described and an evaluation made of the effectiveness of the FRED system. Two specific experiments were performed. First the actual 1971-72 timetable for the Faculty of Business Administration was entered into the system in order to examine the FRED listing facility (timetable print out ready for distribution, list of course conflicts, and individual professor's schedules). Second a complete timetable was built for the Faculty of Business Administration.

Listing the Timetable

The existing 1971-72 timetable was read in by utilizing cards. This required the punching of 169 cards each with the format course name, course section, course term, days the course is taught, beginning time, ending time, professor's name, professor's code, estimated enrollment, sequence number, and course code. A listing of the complete timetable was then requested from a terminal. This required 20 minutes terminal time. Figure 10 shows a portion of the timetable as printed.

A listing of course conflicts is provided by the

program following a listing of the timetable. This was a conflict free timetable, the request matrix having been taken from the actual 1971-72 timetable, and therefore no direct conflicts appeared in the listing. There were, however, 104 indirect conflicts. Figure 11 presents the conflicts identified. Most of these conflicts were insignificant involving only a few students in multi-section courses, but some showed a large number of students enrolled in both courses suggesting potentially important conflicts, e.g. Marketing 418 with Business 304 as well as Management Science 357.

FIGURE 10

SAMPLE LIST OF 1971-72 TIMETABLE

FACULTY OF BUSINESS ADMINISTRATION AND COMMERCE

1971-72 TIME TABLE

COURSE	SECTION	TERM	DAY	TIME		INSTRUCTOR
BUS566		1	M	1400	1650	MUMEY
BUS567		2	M	1400	1650	TINIC
FIN304	A1	1	TT	800	920	SINGH
FIN504	A2	1	TT	1100	1220	SINGH
FIN304	A3	1	MWF	1100	1150	SWIMMER
FIN304	B1	2	TT	800	920	ROLLINS
FIN304	B2	2	TT	1100	1220	GRANT
FIN304	B3	2	MWF	800	850	SWIMMER
FIN401		2	MWF	800	850	SINGH
FIN411	A1	2	TT	1530	1650	GRANT
FIN411	B1	2	TT	930	1050	GRAUL
FIN421	A1	1	MWF	1600	1650	MCFADYEN
FIN421	B1	2	TT	1530	1650	MELNYK
FIN432	A1	1	TT	1400	1520	ROLLINS
FIN432	B1	2	TT	1230	1350	HOSKINS
FIN562		1	F	1300	1550	GRAUL
FIN563		2	F	1300	1550	WEST
I.R304	A1	1	MWF	800	850	SWIMMER
I.R304	A2	1	MWF	1000	1050	LAWSON
I.R304	A3	1	MWF	1100	1150	NEDD
I.R304	B1	2	MWF	1100	1150	B WILLIAMS
I.R304	B2	2	TT	800	920	HAMEED
I.R401		1	TT	1530	1650	MELNYK
I.R411		1	MWF	1500	1550	B WILLIAMS
I.R412		2	MWF	1500	1550	B WILLIAMS
I.R414		2	MWF	1300	1350	B WILLIAMS
I.R421		1	T	1530	1820	SHIHADEH
I.R422		2	T	1530	1820	RASMUSSEN
I.R563		1	F	900	1150	B WILLIAMS
I.R565		2	F	900	1150	SHIHADEH
MKT304	A1	1	MWF	900	950	WEERES
MKT304	A2	1	MWF	1100	1150	WEERES
MKT304	A3	1	MWF	1400	1450	WEERES
MKT304	B1	2	MWF	900	950	WEERES
MKT304	B2	2	MWF	1100	1150	WEERES
MKT304	B3	2	MWF	1400	1450	WEERES
MKT402		2	MWF	900	950	DUNN
MKT403	A1	1	MWF	1400	1450	YOUNG
MKT403	A2	1	MWF	1200	1250	YOUNG
MKT405		1	MWF	1000	1050	DUNN
MKT407	B1	2	TT	1230	1350	YOUNG
MKT411		1	MWF	1100	1150	HUGSTAD
MKT414		1	MWF	1300	1350	DUNN
MKT418		2	TT	1230	1350	HUGSTAD
MKT419		2	MWF	1200	1250	MIRUS

FIGURE 11

SAMPLE LIST OF CONFLICTS 1971-72

```
I.R412    IS IN INDIRECT CONFLICT WITH BUS333  F3   2 STUDENTS ARE INVOLVED
I.R412    IS IN INDIRECT CONFLICT WITH BUS333  F4   2 STUDENTS ARE INVOLVED
1.R412    IS IN INDIRECT CONFLICT WITH MSC357  B3  14 STUDENTS ARE INVOLVED
ACT408    IS IN INDIRECT CONFLICT WITH BUS304  B2   8 STUDENTS ARE INVOLVED
ACT408    IS IN INDIRECT CONFLICT WITH BUS312  B2   2 STUDENTS ARE INVOLVED
ACT408    IS IN INDIRECT CONFLICT WITH FIN304  B1   4 STUDENTS ARE INVOLVED
ACT408    IS IN INDIRECT CONFLICT WITH I.R304  B2   4 STUDENTS ARE INVOLVED
MKT418    IS IN INDIRECT CONFLICT WITH ACT204  G1   8 STUDENTS ARE INVOLVED
MKT418    IS IN INDIRECT CONFLICT WITH BUS304  B3  24 STUDENTS ARE INVOLVED
MKT418    IS IN INDIRECT CONFLICT WITH BUS307  B6   2 STUDENTS ARE INVOLVED
MKT418    IS IN INDIRECT CONFLICT WITH FIN432  B1   4 STUDENTS ARE INVOLVED
MKT418    IS IN INDIRECT CONFLICT WITH MSC357  B4  48 STUDENTS ARE INVOLVED
MKT418    IS IN INDIRECT CONFLICT WITH MKT407  B1   4 STUDENTS ARE INVOLVED
ACT412    IS IN INDIRECT CONFLICT WITH BUS307  B5   4 STUDENTS ARE INVOLVED
ACT412    IS IN INDIRECT CONFLICT WITH BUS402  A3   4 STUDENTS ARE INVOLVED
MSC411    IS IN INDIRECT CONFLICT WITH BUS402  A3   8 STUDENTS ARE INVOLVED
I.R422    IS IN INDIRECT CONFLICT WITH BUS333  D4   4 STUDENTS ARE INVOLVED
I.R422    IS IN INDIRECT CONFLICT WITH FIN411  A1  14 STUDENTS ARE INVOLVED
I.R422    IS IN INDIRECT CONFLICT WITH FIN421  B1  14 STUDENTS ARE INVOLVED
I.R422    IS IN INDIRECT CONFLICT WITH BUS333  E4   4 STUDENTS ARE INVOLVED
```

It was interesting to note that the conflict listing brought out that some indirect conflicts regarded as inconsequential by the scheduler were in fact very real such as the above where a fourth year marketing course overlapped with two third year required courses.

The 104 represents the number of indirect conflicts between courses, i.e. courses with the same participants, not the number of students who had conflicts. This latter number could conceivably be just as important as it is not apparent which is more serious - several students with a few conflicts each, or a few students with several conflicts each. The question of establishing a norm of what constitutes an acceptable level of these two types of conflicts in a timetable is an important one. This issue, however, should really be addressed by the Faculty administrator rather than the scheduler.

Having completed the listing of the total timetable, as well as the conflicts identified, the next step was to list the timetable by professor. This was done to examine the schedule of each individual professor. A typical professor schedule is illustrated in figure 12.

There are two reasons for presenting professor's schedules. Firstly, the scheduler needs to consider the balance of each professor's load between terms, weekdays and hours of the timetable. Secondly, he needs to generate the timetable in a form useful for distribution to the staff.

If the schedules of some professors appeared unacceptable - imbalanced loads between terms, inconvenient hours of the day or too few days free from teaching - then the timetable would be modified by means of the CHANGE routine (see chapter 4). This would also be the case if the level of conflicts were unacceptable, or if the courses were imbalanced over the available time slots or between terms.

FIGURE 12

SAMPLE LIST OF 1971-72 PROFESSOR TIMETABLE

MCFADYEN

FIRST TERM					SECOND TERM				
BUS312	A2	MWF	1000	1050	BUS450		MWF	1400	1450
BUS312	A3	MWF	800	850					
FIN421	A1	MWF	1600	1650					

MCGEACHY

FIRST TERM	SECOND TERM

MELYNK

FIRST TERM					SECOND TERM				
BUS310	A1	TT	930	1050	BUS310	B1	TT	930	1050
I.R401		TT	1530	1650	FIN421	B1	TT	1530	1650

Building a Timetable

The next phase was to build a timetable. Since the complete process of constructing a timetable is quite tedious and the purpose here simply is to illustrate and test the system the timetable was developed by starting from a partial timetable and finishing the task. This was preferred to starting at the beginning and carrying the illustration only part way through the process. In order to quickly reach a partial timetable all the required courses for the Bachelor of Commerce degree were entered into the system via cards leaving the optional courses to be scheduled by means of FRED. The partial timetable so achieved was listed by time period to provide information about what time periods were good candidates for more courses. This required about 45 minutes terminal time. This listing feature takes twice as long as the previous listing because more typing is required.

Having identified the time periods with room for additional courses the non-required courses were entered into these times. Courses with generally larger enrollments were scheduled first. After a new course has been entered the program immediately displays the conflicts created. The scheduler can then remove that course, or one of the other courses involved in the conflict, from that time period using the CHANGE routine. An updated listing of the timetable was requested after about ten courses had been

entered to identify additional time period candidates. This process was repeated until all courses had been entered. After having made the modifications required to avoid conflicts or to improve professor schedules the timetable was complete.

It should be recognized that it is much faster to enter courses via cards than via the terminal. Hence it is generally quicker to enter a complete timetable, whether feasible or not, at the outset and use the terminal only for modifications. Even if the initial timetable is not a particularly good one alterations are easily made as the CHANGE routine is quite efficient. Thus this procedure would be preferred to stepwise creation of a timetable from the terminal.

Overall Considerations

From the experience gained during the terminal sessions, and from discussion afterward with McGeachy, several things were learned about the FRED system operating under real conditions. First, the FRED system is an economic tool for developing a timetable. For a complete run and listing the cost was approximately $8.00 (based on $300.00 per CPU hour). This is a relatively inexpensive way to examine the timetable. Second, information provided is helpful - a current report of teaching loads, an identification of conflicts as well as time periods utilized. This information was not readily available to the scheduler from the manual method. Third, by readily

presenting the information needed for the next step the system does reduce the scheduler's decision time. Fourth, the job can be resumed after a break from the point where it was left wihout time lost in getting back to the task.

At present the system reports only conflicts involving single section courses. The conflict feature is therefore only valuable as long as there are single section courses in the timetable. A modification could be made to the program to cover also conflicts between multi-section courses. However, when providing additional information it must be remembered that too much data may be difficult for the scheduler to digest and could actually reduce the efficiency of the system.

For purposes of practical application some modifications in order to personalize the system could be made. The FRED system was written for a person unfamiliar with the computer. Once a person becomes familiar with the system instructions to the user could be considerably shortened. Also, if a light-pen and a cathode ray tube were used the process would be more efficient.

<u>Conclusions and Proposed Future Research</u>

The aim of the research undertaken has been to provide a useful aid to the human scheduler in the area of data manipulation and timetable checking. The system designed provides flexibility to handle timetable modifications as well as maintains information for evaluating the resulting

timetable. Additional features can be built in as the criteria of timetable goodness evolve.

Some modification to the system which should be considered are the following. Some form of sectioning could be implemented to assign students to sections. Also typical student course patterns could be identified and a special attempt made to accommodate these in the scheduling process. A third area is that of room assignment, which is currently being handled by the Registrar's Office. The utilization of classrooms is an important aspect from the overall university perspective. Because there is a finite number of rooms of given seating capacities available during any one period the timetable must be built under this constraint. Proper scheduling to utilize the available classrooms is desirable and this aspect could be incorporated into the system. At present the Faculty of Business has no control over the assignment of courses to rooms and this aspect was therefore left out.

The system could also be modified to make recommendations about appropriate changes that might be made to the timetable, e.g., recommendations regarding where best to place a particular course. This would remove some of the guess work presently required of the human scheduler and allow the machine a larger role in the development of a timetable.

The aim of the research has been to provide a usable and useful aid to the human scheduler in the area of data

manipulation and timetable checking, an aid which would in fact be used. The FRED system does meet these characteristics and it is hoped that the system will find favor with the scheduler in the Faculty of Business Administration and enjoy use in the years to come.

APPENDIX 1

SAMPLE TERMINAL SESSION

Following is a listing from terminal session using all features of the FRED system. For clarification, comments have been inserted between the various functions.

This routine lists the Table of Contents.

```
WELCOME TO THE FRED SCHEDULER.  THE EXISTING TIME TABLE HAS BEEN LOADED

TYPE 0 FOR THE TABLE OF CONTENTS -OR- THE PROCEDURE NUMBER DESIRED
0

        TABLE OF CONTENTS

1=LIST THE CURRENT TIME TABLE
2=CHECK FOR CONFLICTS
3=LIST ALL PROFESSORS TIME TABLES
4=DELETE A COURSE
5=CHANGE A COURSE TIME
6=REASSIGN A PROFESSOR
7=INSERT A NEW COURSE
8=SAVE CURRENT TIMETABLE
9=CHANGE THE REQUEST MATRIX
10=STOP THE PROGRAM
```

This procedure lists the timetable in alphabetical order.

```
PLEASE TYPE THE PROCEDURE NUMBER YOU DESIRE
1

TYPE 1 :-LIST BY PAGE,
     2 :-LIST BY COURSE,
     3 :-LIST BY TIME,
     4 :-RETURN.
1
```

FACULTY OF BUSINESS ADMINISTRATION AND COMMERCE

TIME TABLE

COURSE	SECTION	TERM	DAY	TIME		INSTRUCTOR
ACT304	A1	1	MWF	1100	1150	COHOON
ACT304	A2	1	R	1230	1420	COHOON
ACT304	B1	2	MWF	1100	1150	COHOON
ACT304	B2	2	R	1230	1420	COHOON
ACT562		2	T	1400	1650	COHOON
BUS312	A2	1	MWF	1000	1050	MCFADYEN
BUS323	A1	1	MWF	900	950	EARL
BUS323	A2	1	MWF	1100	1150	Earl
BUS357	A1	2	TT	1000	1230	WINTERS

This procedure lists the timetable for a specific course.

```
TYPE 1 :-LIST BY PAGE,
     2 :-LIST BY COURSE,
     3 :-LIST BY TIME,
     4 :-RETURN.
2

ENTER COURSE NAME
act304

        FACULTY OF BUSINESS ADMINISTRATION AND COMMERCE

                          TIME TABLE

 COURSE   SECTION   TERM   DAY      TIME      INSTRUCTOR

ACT304       A1       1    MWF   1100 1150      COHOON
ACT304       A2       1    R     1230 1420      COHOON
ACT304       B1       2    MWF   1100 1150      COHOON
ACT304       B2       2    R     1230 1420      COHOON

TYPE 0 FOR THE TABLE OF CONTENTS -OR- THE PROCEDURE NUMBER DESIRED
1
```

This procedure lists the timetable by time period.

```
TYPE 1 :-LIST BY PAGE,
     2 :-LIST BY COURSE,
     3 :-LIST BY TIME,
     4 :-RETURN.
3

ENTER DATE,TERM,BEGIN TIME SEPARATE BY COMMA

mwf,1,1100



              FACULTY OF BUSINESS ADMINISTRATION AND COMMERCE

                               TIME TABLE

COURSE   SECTION   TERM   DAY     TIME        INSTRUCTOR

ACT304     A1        1    MWF   1100 1150       COHOON
BUS323     A2        1    MWF   1100 1150       EARL

TYPE 1 :-LIST BY PAGE,
     2 :-LIST BY COURSE,
     3 :-LIST BY TIME,
     4 :-RETURN.
4
```

Procedure 2 prints out the conflicts.

```
TYPE 0 FOR THE TABLE OF CONTENTS -OR- THE PROCEDURE NUMBER DESIRED.
2

    ACT304  A1 IS IN DIRECT CONFLICT WITH BUS323  A2   26   STUDENTS ARE INVOLVED
    BUS323  A2 IS IN DIRECT CONFLICT WITH ACT304  A1   26   STUDENTS ARE INVOLVED
```

Procedure 3 lists the professors' timetables. This portion lists all professors.

```
TYPE 0 FOR THE TABLE OF CONTENTS -OR- THE PROCEDURE NUMBER DESIRED.
3

DO YOU WISH TO SEE ALL THE PROFESSORS TIME TABLES?  1=YES,2=NO
1

COHOON
                       FIRST TERM                      SECOND TERM
         ACT304  A1  MWF   1100 1150      ACT304  B1  MWF   1100 1150
         ACT304  A2  R     1230 1420      ACT304  B2  R     1230 1420
                                          ACT562      T     1400 1650
EARL
                       FIRST TERM                      SECOND TERM
         BUS323  A1  MWF    900  950
         BUS323  A2  MWF   1100 1150
GRANT
                       FIRST TERM                      SECOND TERM
MCFADYEN
                       FIRST TERM                      SECOND TERM
         BUS312  A2  MWF   1000 1050
```

This procedure lists only those professors requested.

```
WINTERS
                          FIRST TERM                        SECOND TERM

TYPE 0 FOR THE TABLE OF CONTENTS -OR- THE PROCEDURE NUMBER DESIRED.
3

DO YOU WISH TO SEE ALL THE PROFESSORS TIME TABLES?  1=YES,2=NO
2
PLEASE INSERT PROFESSORS NAMES SEPARATED BY COMMAS
earl,winters

EARL

                          FIRST TERM                        SECOND TERM

               BUS323  A1  MWF   900  950

               BUS323  A2  MWF  1100 1150

WINTERS

                          FIRST TERM                        SECOND TERM

                                                 BUS357  A1  TT   1000 1230
```

The procedure deletes a course. The professor's timetable is listed to prove it has been deleted.

```
TYPE 0 FOR THE TABLE OF CONTENTS -OR- THE PROCEDURE NUMBER DESIRED.
4

TYPE 1 - TO DELETE A COURSE,
     2 - RETURN.
1

ENTER COURSE NAME AND SECTION YOU WISH TO DELETE.
bus323,a2

TYPE 1 - TO DELETE A COURSE,
     2 - RETURN.
2

TYPE 0 FOR THE TABLE OF CONTENTS -OR- THE PROCEDURE NUMBER DESIRED.
3

DO YOU WISH TO SEE ALL THE PROFESSORS TIME TABLES?  1=YES,2=NO
2
PLEASE INSERT PROFESSORS NAMES SEPARATED BY COMMAS
earl

      EARL
                         FIRST TERM                        SECOND TERM

              BUS323  A1  MWF    900 950
```

This procedure changes the time or days of an existing course.

```
TYPE 0 FOR THE TABLE OF CONTENTS -OR- THE PROCEDURE NUMBER DESIRED.
5

INSERT COURSE NAME AND SECTION SEPARATED BY A COMMA
bus357,a1

INSERT TERM,DAYS AND TIME YOU WISH TO MOVE THE COURSE TO ALL SEPARATED BY COMMAS
2,mwf,1400,1450

FOLLOWING IS A LIST OF COURSES TAUGHT BY PROFESSOR WINTERS
BUS357 A1  TT  1000 1230
TYPE 1 - CHANGE ANOTHER COURSE
     2 - CHANGE THE SAME COURSE
     3 - PERFORM THE CHANGE
     4 - RETURN.
3

TYPE 1 - CHANGE ANOTHER COURSE
     2 - CHANGE THE SAME COURSE
     3 - PERFORM THE CHANGE
     4 - RETURN.
4
```

This procedure reassigns the professor who is teaching a course.

```
TYPE 0 FOR THE TABLE OF CONTENTS -OR- THE PROCEDURE NUMBER DESIRED.
6

INSERT THE COURSE NAME AND SECTION FOR WHICH YOU WISH TO REASSIGN PROFESSORS
act304,a1

PROFESSOR COHOON          NOW REACHES THAT COURSE

INSERT THE NAME OF PROFESSOR YOU NOW WISH TO TEACH THIS COURSE
grant

THE PROFESSOR HAS BEEN REASSIGNED
```

This procedure inserts a new course into the timetable.

```
TYPE 0 FOR THE TABLE OF CONTENTS -OR- THE PROCEDURE NUMBER DESIRED.
7

PLEASE TYPE COURSE NAME,SECTION,TERM,DAY,BEGIN TIME,END TIME,PROF,EXPECTED
ENROLLMENT, AND SEQUENCE NUMBER AND COURSE CODE:  ALL SEPARATED BY COMMAS

bus323,al,1,tt,930,1050,earl,10,5

TYPE 1 - INSERT ANOTHER COURSE
     2 - RETURN
2

TYPE 0 FOR THE TABLE OF CONTENTS -OR- THE PROCEDURE NUMBER DESIRED.
3

DO YOU WISH TO SEE ALL THE PROFESSORS TIME TABLES? 1=YES,2=NO
2
PLEASE INSERT PROFESSORS NAMES SEPARATED BY COMMAS
earl


   EARL
                 FIRST TERM                          SECOND TERM

          BUS323  A1  TT    930 1050
```

Procedure 9 is used to modify the request matrix.

Procedure 8 writes the new timetable out on a file.

Procedure 10 signs off the FRED system.

```
TYPE 0 FOR THE TABLE OF CONTENTS -OR- THE PROCEDURE NUMBER DESIRED.
9

ENTER ROW,COLUMN AND ENROLLMENT
(RETURN IF ROW NUMBER = 0)
12,14,8

ENTER ROW,COLUMN AND ENROLLMENT
(RETURN IF ROW NUMBER = 0)
#

TYPE 0 FOR THE TABLE OF CONTENTS -OR- THE PROCEDURE NUMBER DESIRED.
8

DO YOU WISH TO SAVE THE CORRECT TIME TABLE?  1=YES,2=NO
1
SOVE       - SPECIFY FDNAME OR SEND END-OF-FILE
aa

TIMETABLE SAVED

TYPE 0 FOR THE TABLE OF CONTENTS -OR- THE PROCEDURE NUMBER DESIRED.
10
DO YOU WISH TO SAVE THE CURRENT TIME TABLE? 1=YES,2=NO
2
GOOD-BYE FROM THE FRED SCHEDULER
```

APPENDIX 2

VARIABLE DICTIONARY AND PROGRAM LISTING

AA	Temporary Work Storage
AGAIN	Statement Label Constant
ANSWER	Procedure None
B1	Temporary Work Storage
BEGIN	Statement Label Constant
BEGIN-TIME	Variable for starting time of a course
C1	Statement label constant
C2	Statement label constant
C-Code	Code for a course
C-DAY	Day a course is held
C-ENROLLMENT	Enrollment in a course
C-MATRIX	Procedure name
C-NAME	Name of course
C-PROF-CODE	Code for professcr teaching a course
C-SEC	Section number of the course
C-SEQ-NO	Sequence number of the course
C-TABLE	Table of all course
C-TERM	Term in which course is offered
CARD1	Character string input
CARD2	Character string input
CARDS	Parameter
CHANGE	Procedure name
PROF	Statement label constant
CHNG	Statement label constant
CLASS-DATE	Day course is offered
CLASS-NAME	Name of course
CLASS-PER	Period a course is taught

CLASS-SECTION	Section number of a course
CLASS-TIME	Time a course is offered
CLASSES	(8,2) in Prof-Time(*), structure CODE
CIS-CODE	Course code
COL	Column of request matrix to be changed
CON-TYPE	Type of conflict
CONFL	Procedure name
CONS	Temporary Work Storage
COURSE	File, external
CSW	Parameter
CSW	Temporary Work Storage
D1	Temporary Work Storage
D1	Statement label constant
D2	Temporary Work Storage
D3	Temporary Work Storage
D33	Temporary Work Storage
D34	Temporary Work Storage
D4	Temporary Work Storage
D5	Temporary Work Storage
DELETE	Procedure Name
DUMMY	Temporary Work Storage Structure
K	Ending Time Block
ECHAR	Temporary Work Storage
END-TIME	End time of a course
ENR	Temporary Work Storage
ENROLLMENT	Enrollment of the course
F	Temporary Work Storage
F1	Statement label constant

F2	Statement label constant
FCHAR	Temporary Work Storage
FILE-DAY	Statement label constant
FILE-DAY	Statement label constant
FILE-NEXT	Statement label constant
FILE-PROF	Procedure name
FILE-TABLE	Statement label constant
FINIS	Statement label constant
FIRST	Statement label constant
FIVE	Statement label constant
FOUND	Statement label constant
FOUND	Statement label constant
FOUND	Statement label constant
FOUND	Statement label constant
FOUND	Statement label constant
G1	Statement label constant
H	Label
I	Temporary Work Storage
I1	Temporary Work Storage
I2	Temporary Work Storage
IALPHA	Temporary Work Storage
IC	Temporary Work Storage
II	Temporary Work Storage
III	Temporary Work Storage
III	Temporary Work Storage
IN-FILE	File, externa, record, input
INDEX	Generic, built-in function
INSERT	Procedure name

INVALID	Statement label constant
NC-COURSES	Course time
NC-OF-COURSES	Course time
NC-PROFS	Total number of professors on faculty
J	Temporary Work Storage
J1	Temporary Work Storage
J2	Temporary Work Storage
J4	Temporary Work Storage
J5	Temporary Work Storage
J6	Temporary Work Storage
JUMP	Parameter
JUMP	Temporary Work Storage
K	Parameter
K	Time block a course is offered in
K100	Temporary Work Storage
K200	Temporary Work Storage
KK	Temporary Work Storage
KL	Temporary Work Storage
KT	Temporary Work Storage
L	Temporary Work Storage
L1	Temporary Work Storage
L1	Statement label constant
L2	Temporary Work Storage
L2	Statement label constant
L2	Statement label constant
L2	Statement label constant
L3	Statement label constant
L3	Statement label constant

L3	Statement label constant
L 4	Statement label constant
L4	Statement label constant
L5	Statement label constant
L5	Statement label constant
L6	Statement label constant
L6	Statement label constant
L7	Statement label constant
L7	Statement label constant
L-CODE	Listing Code by page, etc
L-END	Statement label constant
L-TABLE	Procedures name
LCHAR	Temporary Work Storage
LENGTH	Generic, built-in function
LINE	Counter
LM	course time
M	Temporary Work Storage
M1	Temporary Work Storage
MAIN	Procedure name
MAX-CLASS	Maximum number of concurrent course
desired	
MESSAGE	Character string for printing messages
N	Temporary Work Storage
N1	Temporary Work Storage
N2	Temporary Work Storage
N3	Temporary Work Storage
N-PROF	Professor name
N-PROF	Professor name

N-PROFS	Number of professors
NAME	Course name
NEW-PROF	Statement label constant
NEXT-COURSE	Statement label constant
NEXT-P	Statement label constant
NEXT-SEC	Statement label constant
NO-CLASS	Number of concurrent classes
NO-CONF	Statement label constant
OUT	Statement label constant
P1	Pointer to timetable
P2	Pointer to timetable
P3	Pointer to timetable
P-CCDE	Professor code
P-DUMMY	Temporary Work Storage
P-HDG	Statement label constant
P-P-CODE	Professor co
PHASE-2	Statement label constant
PHASE-2	Statement label constant
PRINT	Statement label constant
PROF	Professor name
PROF-CODE	Professor code
PROF-TIME-TABLE	Structure containing professor timetables
PROFESSOR	Professor name in prof time table
PROFS	File, external
R-L-CODE	Statement label constant
R-READ	Procedure name
READ	Statement label constant
READ-C-T	Statement label constant

REAL-PROF	Statement label constant
REASSGN	Procedure name
REQM	File, external
REQUEST-MATRIX	Matrix containing students requests
ROW	Row of request-matrix to be changed
S-K	Starting Time Period
SAVE	Procedure name
SEC-ASS	Statement label constant
SECOND	Statement label constant
SECTION	Course section in time table
SELECT	Procedure name
SEQ-NO	Sequence number of 6 course
SIX	Statement label constant
SM	Course time
SORT-TABLE	Procedure name
SOVE	File,external,output
START-LIST	Statement label constant
SUBSTR	Generic, built-in function
SW	Temporary Work Storage
T	Course time
T1	Course time
T2	Course time
T3	Course time
T-COURSE	Temporary Work Storage
T-DATE	Temporary Work Storage
T-PER	Temporary Work Storage
T-TITLE	Temporary Work Storage
TEMP-CODE	Temporary Work Storage

TEMP-TERM	Temporary Work Storage
THREE	Statement label constant
TI	Temporary Work Storage
TIME	Course start and end times
TIME-PERIOD	Procedure name
TIME-TABLE	Master structure holding timetable
TITLE	Course name and section in vc(300)
TRM	Course term in time table
TT	Statement label constant
TYPE-CODE	Statement label constant
TYPEAGAIN	Statement label constant
TYPEAGAIN	Statement label constant
VC	Vector of course sorted alphabeticaly
NO.	Number of course in VC
WC	Combination of days available to teach
course	
WD	Numeric representation of WC
WHAT	Statement label constant
WORD	Parameter
WORDS	Parameter
WORDS	Answers from the terminal
WRONG	Statement label constant
X1	Parameter
X2	Parameter
X3	Parameter
ZZ	Statement label constant

```
        MAIN:PROCEDURE OPTIONS(MAIN);                                                          PAGE      2

 STMT LEVEL NEST
   1                 MAIN:PROCEDURE OPTIONS(MAIN);
   2     1                DCL WD(10) CHAR(3) INIT('MWP','MW ','MF ','WF ',
                                            'M  ','W  ','F  ','TT ','T  ','R  ');
   3     1                DCL WC(10) FIXED BIN(15) INIT(0,-3,-2,-1,1,2,3,0,1,2);
   4     1                DCL (X1,X2,X3,CSW) FIXED BIN(15);
   5     1                DCL  MAX_CLASS   FIXED BIN(31),
                               #_COURSES     FIXED BIN(31),
                               #_PROFS       FIXED BIN(31);

   6     1                DCL  P_CODE  FIXED DEC(2),
                               P_NAME  CHAR(15) VARYING,
                               C_NAME  CHAR(6),
                               C_SEC   CHAR(2),
                               C_TERM  FIXED BIN(15),
                               TEMP_TERM FIXED BIN(15),
                               C_DAY   CHAR(3),
                               BEGIN_TIME   FIXED DEC(4),
                               END_TIME     FIXED DEC(4),
                               C_PROF_CODE    FIXED BIN(31),
                               TEMP_CODE FIXED BIN(31),
                               C_ENROLLMENT     FIXED DEC(3),
                               C_SEQ_NO       FIXED BIN(31),
                               N_PROF CHAR(15),
                               C_CODE   FIXED DEC(3);

   7     1                DCL (I,M,JUMP,L,K,AA,B1,F,I1,J1,KK,III,IC) FIXED BIN(15),
                               LM  FIXED BIN(31) INIT(0),
                               C_TABLE(100) CHAR(6),
                               NO_CLASS(16,2)  FIXED BIN(31) INIT((32)0);

   8     1                DCL IN_FILE FILE  RECORD INPUT,
                              P_DUMMY CHAR(15),
                              CARDS CHAR(80) VARYING,
                              WORDS CHAR(20) VARYING,
                              MESSAGE CHAR(100) VARYING;
   9     1                    MESSAGE =
                              'TYPE 0 FOR THE TABLE OF CONTENTS -OR- THE PROCEDURE NUMBER'
                              ||' DESIRED.';
  10     1               DCL IALPHA FIXED BIN(31);
  11     1               DCL (SW,SM) BIT(1) INIT('0'B);
  12     1               DCL VC_# FIXED BIN(15) INIT(0);
  13     1               DCL 1 DUMMY,
                                  2  TI CHAR(8),
                                  2  T1 FIXED BIN(15),
                                  2  T2  FIXED BIN(15),
                                  2  T3  FIXED BIN(15);

  14     1               DCL 1 VC(300),
                               2  TITLE CHAR(8),
                               2  P1     FIXED BIN(15),
                               2  P2     FIXED BIN(15),
                               2  P3     FIXED BIN(15);
  15     1                  GET FILE(COURSE) DATA (MAX_CLASS,#_COURSES,#_PROFS);

  16     1                 IF (MAX_CLASS * #_COURSES * #_PROFS) < 1 THEN
  17     1                    DO;
```

```
         MAIN:PROCEDURE OPTIONS(MAIN);                                                   PAGE    3

 STMT LEVEL NEST

   18     1    1            PUT SKIP EDIT('INVALID MAX. CLASS/NO. OF COURSES',
                                      '/NO. OF PROFS, JOB CANCELLED.') (A);
   19     1    1            RETURN;
   20     1    1            END;

   21     1              BEGIN;
   22     2           ON ENDFILE(REQM) BEGIN;  GO TO READ_PROF;  END;
   26     2              ON ENDFILE(PROFS) BEGIN;
   28     3                 GO TO READ;
   29     3                 END;

   30     2              ON ENDFILE(COURSE) BEGIN;
   32     3                 GO TO PHASE_2;
   33     3                 END;

   34     2              DCL H(10) LABEL;

   35     2                   DCL 1  TIME_TABLE(16,20,2),
                                       2  NAME  CHAR(6),
                                       2  SECTION   CHAR(2),
                         2 T_DATE FIXED BIN(15),
                         2 T_PER  FIXED BIN(15),
                                       2  TRM FIXED DEC(1),
                                       2  CODE      FIXED BIN(15),
                                       2  PROF      CHAR(15),
                                       2  PROF_CODE FIXED BIN(15),
                                       2  SEQ_NO    FIXED BIN(15),
                                       2  ENROLLMENT  FIXED BIN(31),
                                       2  CLS_CODE    FIXED BIN(15),
                                       2  TIME       CHAR(14);
   36     2                  DCL  REQUEST_MATRIX(#_COURSES,#_COURSES)   FIXED BIN(31);
   37     2                  DCL   1 PROF_TIME_TABLE(#_PROFS),
                                       2  PROFESSOR     CHAR(15),
                                       2  #_OF_CLASSES (2)  FIXED BIN(15)    ,
                                       2  CLASSES (8,2),
                                          3  CLASS_NAME   CHAR(6),
                                          3  CLASS_SECTION  CHAR(2),
                                          3  CLASS_TIME  CHAR(14),
                         3 CLASS_DATE FIXED BIN(15),
                         3 CLASS_PER  FIXED BIN(15),
                                          3  S_K  FIXED DEC(4),
                                          3  E_K  FIXED DEC(4);

   38     2              GET FILE(REQM) EDIT(REQUEST_MATRIX) (F(5));
   39     2              #_OF_CLASSES = 0;
   40     2              CLASS_NAME=' ';
   41     2              CLASS_SECTION = ' ';
   42     2              CLASS_TIME = ' ';
   43     2              C_TABLE = ' ';
   44     2              S_K = 0;
   45     2              E_K = 0;
   46     2        READ_C_T: GET FILE(REQM) SKIP EDIT(J,C_NAME)(X(3),F(2),X(1),A(6));
   47     2           IF C_TABLE(J) ¬= ' ' THEN
   48     2              DO; PUT SKIP EDIT('DUPLICATED COURSE CODE',J,C_NAME)
```

```
STMT LEVEL NEST

                                         (A,F(5),X(3),A(6));
  50    2    1            GO TO READ_C_T;
  51    2    1            END;
  52    2               C_TABLE(J) = C_NAME;
  53    2               GO TO READ_C_T;
  54    2            READ_PROF:
                          GET SKIP FILE(PROFS) EDIT (P_CODE) (F(2))
                                                    (P_NAME) (A(15));
  55    2                 PROFESSOR(P_CODE) = P_NAME;
  56    2                 GO TO READ_PROF;
  57    2            READ:
                          GET FILE(COURSE) EDIT (C_NAME) (A(6))
                                                (C_SEC) (A(2))
                                                (C_TERM) (F(1))
                                                (C_DAY) (X(1),A(3))
                                                (BEGIN_TIME) (X(2),F(4))
                                                (END_TIME) (X(1),F(4))
                                                (C_PROF_CODE) (X(16),F(2))
                                                (C_ENROLLMENT) (F(3))
                                                (C_SEQ_NO) (F(2))
                                                 (C_CODE) (F(3));
  58    2               IF C_NAME ¬= C_TABLE(C_CODE) THEN GO TO WRONG;
  60    2               IF C_TERM < 1 | C_TERM > 2 THEN GO TO WRONG;
  62    2                 CALL TIME_PERIOD(JUMP,K);
  63    2                 IF JUMP = 0 THEN GO TO FILE_TABLE;
  65    2             WRONG:PUT SKIP EDIT(C_NAME) (A)
                                    (C_SEC) (X(2),A)
                                    (C_TERM) (F(5))
                                    ('  HAS NOT BEEN SCHEDULED') (A);
  66    2                 GO TO READ;
  67    2            FILE_TABLE:
                        CALL FILE_PROF;
  68    2               GO TO READ;
  69    2             PHASE_2: CALL SORT_TABLE;

                  /* THE MATRICES HAVE BEEN LOADED AND SORTED */

  70    2                    PUT SKIP(2);
  71    2                 PUT EDIT ('WELCOME TO THE FRED SCHEDULER.  THE EXISTING ') (A)
                                  ('TIME TABLE HAS BEEN LOADED') (A);
  72    2             WHAT: PUT SKIP(3) EDIT(MESSAGE) (A);
  73    2                 CALL ANSWER;
  74    2                 GO TO ZZ;
  75    2             H(1): CALL L_TABLE;
  76    2                 GO TO WHAT;
  77    2             H(2):
                        DO X3 = 1 TO 2;
  78    2    1          DO X1 = 1 TO 16;
  79    2    2          DO X2 = 1 TO NO_CLASS(X1,X3);
  80    2    3            C_NAME = NAME(X1,X2,X3);
  81    2    3            C_SEC = SECTION(X1,X2,X3);
  82    2    3            C_TERM = X3;
  83    2    3            C_CODE = CODE(X1,X2,X3);
  84    2    3            C_SEQ_NO = SEQ_NO(X1,X2,X3);
```

```
          MAIN:PROCEDURE OPTIONS(MAIN);

 STMT LEVEL NEST

   85     2    3             CALL CONFL(X1,X2,X3);
   86     2    3           END;
   87     2    2           END;
   88     2    1           END;
   89     2               GO TO WHAT;
   90     2          H(3): PUT SKIP EDIT('DO YOU WISH TO SEE ALL THE PROFESSORS ',
                              'TIME TABLES? 1=YES,2=NO') (A,A);
   91     2                 CALL ANSWER;
   92     2                IF IALPHA = 1 THEN DO;
   94     2    1                                DO KK=1 TO #_PROFS;
   95     2    2                                P_DUMMY = PROFESSOR(KK);
   96     2    2                                CALL SELECT(P_DUMMY);
   97     2    2                                END;
   98     2    1                            END;
   99     2                  ELSE DO;
  100     2    1                    PUT EDIT('PLEASE INSERT PROFESSORS NAMES SEPERATED ',
                                         'BY CCMMAS') (A,A);
  101     2    1                    PUT SKIP;
  102     2    1                    READ FILE(IN_FILE) INTO(CARDS);
  103     2    1              NEXT_P: CALL R_READ(CARDS,WORDS);
  104     2    1                    P_DUMMY = WORDS;
  105     2    1                    IF CARDS = '' & WORDS = '' THEN GO TO WHAT;
  107     2    1                    CALL SELECT(P_DUMMY);
  108     2    1                    GO TO NEXT_P;
  109     2    1              END;
  110     2                   GO TO WHAT;
  111     2           H(4): CSW = 0;
  112     2            NEXT_COURSE:
                          PUT SKIP EDIT('TYPE 1 - TO DELETE A COURSE,',
                                        '      2 - RETURN.') (A,SKIP,A);
  113     2               CALL ANSWER;
  114     2               IF IALPHA = 1 THEN DO;
  116     2    1       AGAIN:  PUT SKIP EDIT('ENTER COURSE NAME AND SECTION YOU ',
                                         'WISH TO DELETE.') (A,A);
  117     2    1            PUT SKIP;
  118     2    1            READ FILE(IN_FILE) INTO (CARDS);
  119     2    1            CALL R_READ(CARDS,WORDS);
  120     2    1            C_NAME = WORDS;
  121     2    1            CALL R_READ(CARDS,WORDS);
  122     2    1            C_SEC = WORDS;
  123     2    1            TI = C_NAME||C_SEC;
  124     2    1            DO I = 1 TO VC_#;
  125     2    2              IF TITLE(I) = TI THEN GO TO FOUND;
  127     2    2            END;
  128     2    1            PUT SKIP EDIT('INVALID COURSE NAME OR SECTION, ',
                                         'TRY AGAIN.') (A,A);
  129     2    1            GO TO NEXT_COURSE;
  130     2    1      FOUND:CALL DELETE(I,CSW);
  131     2    1            GO TO NEXT_COURSE;
  132     2    1            END;

  133     2              IF CSW > 0 THEN DO;
  135     2    1                         CALL SORT_TABLE;
  136     2    1                         VC_# = VC_# - CSW;
```

```
        MAIN:PROCEDURE OPTIONS(MAIN);

STMT LEVEL NEST

 137     2    1                      END;
 138     2              GO TO WHAT;
 139     2          H(5): CALL CHANGE;
 140     2                GO TO WHAT;
 141     2          H(6): CALL REASSGN;
 142     2                GO TO WHAT;
 143     2          H(7): III=1;
 144     2                 CALL INSERT(III);
 145     2                 GO TO WHAT;

 146     2         H(8): PUT SKIP EDIT('DO YOU WISH TO SAVE THE CURRECT TIME ',
                                   'TABLE?  1=YES,2=NO') (A,A);
 147     2                 CALL ANSWER;
 148     2                 IF IALPHA = 1 THEN CALL SAVE;
 150     2                 GO TO WHAT;
 151     2          H(9): CALL C_MATRIX;
 152     2                 GO TO WHAT;
 153     2          H(10): PUT EDIT('DO YOU WISH TO SAVE THE CURRENT ',
                                'TIME TABLE? 1=YES,2=NO')  (A,A);
 154     2                  CALL ANSWER;
 155     2                  IF IALPHA = 1 THEN CALL SAVE;
 157     2                  PUT SKIP EDIT ('GOOD-BYE FROM THE FRED SCHEDULER')(A);
 158     2                  PUT SKIP;
 159     2                  RETURN;

 160     2         ZZ:     IF IALPHA = 0 THEN DO ;
 162     2    1           PUT  EDIT ('TABLE OF CONTENTS') (SKIP(3),COLUMN(10), A)
                              ('1=LIST THE CURRENT TIME TABLE') (SKIP(2),A)
                              ('2=CHECK FOR CONFLICTS') (SKIP,A)
                              ('3=LIST ALL PROFESSORS TIME TABLES') (SKIP,A)
                              ('4=DELETE A COURSE') (SKIP,A)
                              ('5=CHANGE A COURSE TIME') (SKIP,A)
                              ('6=REASSIGN A PROFESSOR') (SKIP,A)
                              ('7=INSERT A NEW COURSE') (SKIP,A)
                              ('8=SAVE CURRENT TIMETABLE') (SKIP,A)
                              ('9=CHANGE THE REQUEST MATRIX') (SKIP,A)
                              ('10=STOP THE PROGRAM') (SKIP,A);
 163     2    1     TYPEAGAIN:
                          PUT SKIP EDIT ('PLEASE TYPE THE PROCEDURE')(SKIP(2),A)
                                          ('  NUMBER YOU DESIRE') (A);
 164     2    1              CALL ANSWER;
 165     2    1              END;
 166     2                IF IALPHA = 0 | IALPHA > 10 THEN GO TO TYPEAGAIN;
 168     2                    ELSE GO TO H(IALPHA);
 169     2          ANSWER: PROCEDURE;
 170     3             ON CONVERSION BEGIN;
 172     4                         IALPHA = 100;
 173     4                         GO TO L1;
 174     4                         END;
 175     3          TYPEAGAIN:
                           PUT SKIP;
 176     3                 READ FILE(IN_FILE) INTO(CARDS);
 177     3                 CALL R_READ(CARDS,WORDS);
 178     3                 IALPHA = WORDS;
```

```
          MAIN:PROCEDURE OPTIONS(MAIN);

 STMT LEVEL NEST

  179     3          L1:      IF IALPHA < 0 | IALPHA > 10 THEN
  180     3                         DO; PUT SKIP EDIT('INVALID ANSWER',
                                                   ' TRY AGAIN.') (A,A);
  182     3     1                         GO TO TYPEAGAIN;
  183     3     1                   END;
  184     3                     RETURN;
  185     3                  END ANSWER;
  186     2            SORT_TABLE: PROCEDURE;
  187     3             DCL (I,I1,I2) FIXED BIN(15);
  188     3               I2 = VC_# - 1;
  189     3             PHASE_2:  SM = '0'B;
  190     3               DO I = 1 TO I2;
  191     3     1         I1 = I + 1;
  192     3     1         IF TITLE(I) <= TITLE(I1) THEN GO TO THREE;
  194     3     1         DUMMY = VC(I);
  195     3     1         VC(I) = VC(I1);
  196     3     1         VC(I1) = DUMMY;
  197     3     1         SM = '1'B;
  198     3     1    THREE: END;
  199     3               IF SM = '1'B THEN GO TO PHASE_2;
  201     3               END SORT_TABLE;
  202     2          TIME_PERIOD: PROCEDURE(JUMP,K);
  203     3                   DCL (K,JUMP) FIXED BIN(15);
  204     3                 JUMP=0;
  205     3                 DO IC = 1 TO 7;
  206     3     1           IF C_DAY =WD(IC) THEN DO;
  208     3     2                     K = (BEGIN_TIME - 700)/100;
  209     3     2                     IF K > 9 THEN K = 16;
  211     3     2                     RETURN;
  212     3     2                     END;
  213     3     1          END;
  214     3                DO IC = 8 TO 10;
  215     3     1               IF C_DAY = WD(IC) THEN DO;
  217     3     2                                IF BEGIN_TIME >= 800 THEN K = 10;
  219     3     2                                IF BEGIN_TIME >= 930 THEN K = 11;
  221     3     2                                IF BEGIN_TIME >= 1100 THEN K = 12;
  223     3     2                                IF BEGIN_TIME >= 1230 THEN K = 13;
  225     3     2                                IF BEGIN_TIME >= 1400 THEN K = 14;
  227     3     2                                IF BEGIN_TIMF >= 1530 THEN K = 15;
  229     3     2                                IF BEGIN_TIME >= 1700 THEN K=16;
  231     3     2                                RETURN;
  232     3     2                                END;
  233     3     1             END;
  234     3                   JUMP = 1;
  235     3                   RETURN;
  236     3                END TIME_PERIOD;
  237     2          CONFL: PROCEDURE(X1,X2,X3);
  238     3                DCL (X1,X2,X3,I1,J) FIXED BIN(15),
                               CCN_TYPE CHAR(8);

  239     3                I1 = NO_CLASS(X1,X3);
  240     3                IF C_SEC_NO >= 20 THEN RETURN;
  242     3                DO J = 1 TO I1;
  243     3     1            IF REQUEST_MATRIX(C_CODE,CODE(X1,J,X3)) = 0
```

```
             MAIN:PROCEDURE OPTIONS(MAIN);

STMT LEVEL NEST

244    3    1            THEN GO TO NO_CONF;
245    3    1          IF C_NAME = NAME(X1,J,X3) &
246    3    1             C_SEC = SECTION(X1,J,X3) THEN GO TO NO_CONF;
247    3    1          IF SEQ_NO(X1,J,X3) >= 20 THEN CON_TYPE = 'INDIRECT';
249    3    1                             ELSE CON_TYPE = ' DIRECT ';
250    3    1          PUT SKIP EDIT(C_NAME,C_SEC,' IS IN ',CON_TYPE,
                          ' CONFLICT WITH ',NAME(X1,J,X3),SECTION(X1,J,X3),
                          REQUEST_MATRIX(C_CODE,CODE(X1,J,X3)),
                          '  STUDENTS ARE INVOLVED')
                          (X(10),A,X(2),A,A,A,A,A,X(2),A,F(5),A);
251    3    1       NO_CONF: END;
252    3              RETURN;
253    3            END CONFL;
254    2          SAVE: PROCEDURE;
255    3                DCL SOVE FILE OUTPUT;
256    3            DCL (I,L,M,N) FIXED BIN(15);
257    3            DO I=1 TO VC_#;
258    3    1             L = P1(I);
259    3    1             M = P2(I);
260    3    1             N = P3(I);
261    3    1        PUT SKIP FILE (SOVE) EDIT (NAME (L,M,N))(A(6))
                     (SECTION (L,M,N))(A(2))
                     (N)(F(1))
                     (TIME (L,M,N))(X(1),A(14))
                     (PROF_CODE(L,M,N))(X(16),F(2))
                     (ENROLLMENT (P1(I), P2(I),P3(I)))(F(3))
                     (SEQ_NO (L,M,N))(F(2))
                          (CODE(L,M,N))(F(3));
262    3    1       END;
263    3          PUT SKIP EDIT ('TIMETABLE SAVED') (A);
264    3               PUT SKIP;
265    3          END SAVE;
266    2           L_TABLE: PROCEDURE;
267    3               DCL (I,L,M,N,L_CODE,I1,I2,LINE) FIXED BIN(15),
                           T_COURSE CHAR(6);

268    3               ON CONVERSION BEGIN;
270    4                              GO TO INVALID;
271    4                              END;

272    3            TYPE_CODE:
                       PUT SKIP EDIT('TYPE 1 :-LIST BY PAGE,',
                                     '      2 :-LIST BY COURSE,',
                                     '      3 :-LIST BY TIME,',
                                     '      4 :-RETURN.')
                                    (A,SKIP,A,SKIP,A,SKIP,A);
273    3            R_L_CODE:
                       PUT SKIP;
274    3               READ FILE(IN_FILE) INTO(CARDS);
275    3               CALL R_READ(CARDS,WORDS);
276    3               L_CODE = WORDS;
277    3               IF L_CODE < 1 | L_CODE > 4 THEN
278    3            INVALID:    DO; PUT SKIP EDIT('INVALID LIST CODE') (A);
280    3    1                   GO TO R_L_CODE;
```

```
          MAIN:PROCEDURE OPTIONS(MAIN);

STMT LEVEL NEST

 281    3    1                 END;
 282    3               IF L_CODE = 4 THEN RETURN;
 284    3               LINE = 100;
 285    3               IF L_CODE = 2 THEN
 286    3                  DO; PUT EDIT('ENTER COURSE NAME') (SKIP,A);
 288    3    1             PUT SKIP;
 289    3    1             READ FILE(IN_FILE) INTO(CARDS);
 290    3    1             CALL R_READ(CARDS,WORDS);
 291    3    1             C_NAME = WORDS;
 292    3    1             DO I=1 TO #_COURSES;
 293    3    2          IF C_NAME = C_TABLE(I) THEN GO TO D1;
 295    3    2     C1:     END;
 296    3    1     C2:        PUT EDIT('INVALID COURSE NAME.') (SKIP,A);
 297    3    1                GO TO TYPE_CODE;
 298    3    1     D1:
                        GO TO START_LIST;
 299    3    1          END;

 300    3               IF L_CODE = 3 THEN
 301    3          L1:     DO; PUT EDIT('ENTER DATE,TERM,BEGIN TIME SEPERATE',
                                       ' BY COMMA') (SKIP,A,A);

 303    3    1             PUT SKIP;
 304    3    1             READ FILE(IN_FILE) INTC(CARDS);
 305    3    1             CALL R_READ(CARDS,WORDS);
 306    3    1             C_DAY = WORDS;
 307    3    1             DO I=1 TO 10;
 308    3    2                IF C_DAY = WD(I) THEN GO TO L3;
 310    3    2             END;
 311    3    1     L2:     PUT EDIT('INVALID DATE/TERM/TIME, TRY AGAIN')(SKIP,A);
 312    3    1             GO TO L1;
 313    3    1     L3:     CALL R_READ(CARDS,WORDS);
 314    3    1             IF WORDS = '1' | WORDS = '2' THEN GO TO L4;
 316    3    1     L4:     C_TERM = WORDS;
 317    3    1             CALL R_READ(CARDS,WORDS);
 318    3    1             BEGIN_TIME = WORDS;
 319    3    1             CALL TIME_PERIOD(JUMP,K);
 320    3    1             IF JUMP ¬= 0 THEN GO TO L2;
 322    3    1          END;
 323    3          START_LIST:
                           DO I = 1 TO VC_#;
 324    3    1                L = P1(I);
 325    3    1                M = P2(I);
 326    3    1                N = P3(I);
 327    3    1                IF L_CODE = 1 THEN GO TO L5;
 329    3    1                IF L_CODE = 2 THEN
 330    3    1          DO; T_COURSE = SUBSTR(TITLE(I),1,6);
 332    3    2               IF T_COURSE = C_NAME THEN GO TO L5;
 334    3    2                     IF T_COURSE > C_NAME THEN RETURN;
 336    3    2                  END;
 337    3    1               IF L_CODE = 3 THEN
 338    3    1                  DO; IF C_TERM ¬= N THEN GO TO L7;
 341    3    2                      IF K ¬= L THEN GO TO L7;
 343    3    2                                ELSE GO TO L5;
```

```
        MAIN:PROCEDURE OPTIONS(MAIN);                                              PAGE    10

 STMT LEVEL NEST

 344    3    2                END;
 345    3    1     L5:     IF LINE = 100 THEN GO TO P_HDG;
 347    3    1             IF LINE > 45 THEN
 348    3    1                DO; PUT PAGE EDIT('DO YOU WANT TO CONTINUE?',
                                         ' 1=YES, 2=NO') (A,A);
 350    3    2                  PUT SKIP;
 351    3    2                  READ FILE(IN_FILE) INTO(CARDS);
 352    3    2                  CALL R_READ(CARDS,WORDS);
 353    3    2                  IF WORDS = '1' THEN GO TO P_HDG;
 355    3    2                                 ELSE GO TO TYPE_CODE;
 356    3    2                END;
 357    3    1             GO TO L6;
 358    3    1     P_HDG:  LINE=1;
 359    3    1             PUT PAGE EDIT('FACULTY OF BUSINESS ADMINISTRATION AND',
                                  ' COMMERCE','TIME TABLE','COURSE',
                                  'SECTION','TERM','DAY','TIME','INSTRUCTOR')
                                 (X(15),A,A,SKIP(2),X(30),A,SKIP(2),COL(12),
                                  A,X(2),A,X(2),A,X(3),A,X(5),A,X(5),A);
 360    3    1             PUT SKIP(2);
 361    3    1     L6:     PUT EDIT(NAME(L,M,N),SECTION(L,M,N),N,TIME(L,M,N),PROF(L,M,N))
                               (SKIP,COL(10),A,X(6),A,X(6),F(1),X(5),A,X(4),A);
 362    3    1             LINE = LINE +1;
 363    3    1     L7:     END;

 364    3                GO TO TYPE_CODE;
 365    3              END L_TABLE;
 366    2          SELECT: PROCEDURE(WORD);
 367    3              DCL WORD CHAR(15);
 368    3                DCL (I,KT, KL) FIXED BIN (15);
 369    3                DO I = 1 TO #_PROFS;
 370    3    1           IF WORD = PROFESSOR (I) THEN DO;
 372    3    2           PUT SKIP (3) EDIT (PROFESSOR (I)) (COLUMN(10), A);
 373    3    2           PUT SKIP EDIT ('FIRST TERM', 'SECOND TERM')
                              (COLUMN(30), A, COLUMN(60),A);
 374    3    2           KT = 0;
 375    3    2           KL = 0;
 376    3    2    BEGIN: KT = KT + 1;
 377    3    2           KL = KL + 1;
 378    3    2           PUT SKIP;
 379    3    2           IF KT > #_OF_CLASSES (I,1) THEN GO TO SECOND;
 381    3    2         IF KL > #_OF_CLASSES(I,2) THEN GO TO FIRST;
 383    3    2         PUT SKIP EDIT(CLASS_NAME(I,KT,1)) (COL(20),A)
                              (CLASS_SECTION(I,KT,1)) (X(2),A)
                              (CLASS_TIME(I,KT,1)) (X(2),A)
                              (CLASS_NAME(I,KL,2)) (COL(50),A)
                              (CLASS_SECTION(I,KL,2)) (X(2),A)
                              (CLASS_TIME(I,KL,2)) (X(2),A);
 384    3    2         GO TO BEGIN;
 385    3    2   FIRST:  PUT SKIP EDIT(CLASS_NAME (I, KT, 1)) (COLUMN(20),A)
                                 (CLASS_SECTION (I, KT, 1)) (X(2), A)
                                 (CLASS_TIME (I, KT, 1)) (X(2),A);
 386    3    2           GO TO BEGIN;
 387    3    2   SECOND:   IF KL > #_OF_CLASSES (I,2) THEN RETURN;
 389    3    2               PUT SKIP EDIT(CLASS_NAME(I,KL,2)) (COLUMN(50),A)
```

```
STMT  LEVEL NEST

                                         (CLASS_SECTION (I, KL, 2))  (X(2),A)
                                         (CLASS_TIME (I, KL, 2))  (X(2), A);
 390     3    2               GC TO BEGIN;
 391     3    2               END;
 392     3    1               END;
 393     3               PUT EDIT('PROFESSOR  ') (SKIP(2),A)
                              (WORD) (A)
                              ('DOES NOT EXIST') (A);
 394     3               RETURN;
 395     3             END SELECT;
 396     2          REASSGN: PROCEDURE;
 397     3               DCL (I,J,L,L1,L2,IALPHA) FIXED BIN(15);
 398     3               DCL N_PROFS  CHAR(15);
 399     3               DCL 1 DUMMY,
                              2  D1       CHAR(6),
                              2  D2       CHAR(2),
                              2  D3       CHAR(14),
                         2 D33 FIXED BIN(15),
                         2 D34 FIXED BIN(15),
                              2  D4       FIXED DEC(4),
                              2  D5       FIXED DEC(4);
 400     3               DCL N_PROF CHAR(15);
 401     3               DCL T_TITLE CHAR(8),
                          (N1,N2,N3)  FIXED BIN(15);

 402     3          PRINT:   PUT SKIP EDIT(' INSERT THE COURSE NAME AND SECTION') (A)
                                  (' FOR WHICH YOU WISH TO REASSIGN PROFESSORS') (A);
 403     3               PUT SKIP;
 404     3               READ FILE(IN_FILE) INTC(CARDS);
 405     3               CALL R_READ(CARDS,WORDS);
 406     3               C_NAME = WORDS;
 407     3               CALL R_READ(CARDS,WORDS);
 408     3               C_SEC = WORDS;
 409     3               T_TITLE = C_NAME||C_SEC;
 410     3               DO I = 1 TO VC_#;
 411     3    1               IF T_TITLE = TITLE(I) THEN DO; N1 = P1(I);
 414     3    2                                 N2 = P2(I);  N3 = P3(I);
 416     3    2                                 L1 = PROF_CODE(N1,N2,N3);
 417     3    2                                 L2 = TRM(N1,N2,N3);
 418     3    2                                 GC TO FCUND;
 419     3    2                                 END;
 420     3    1               END;

 421     3               PUT SKIP EDIT('THAT COURSE DOES NOT EXIST') (A);
 422     3               GO TO PRINT;

 423     3          FOUND: PUT SKIP EDIT('PROFESSOF ') (A)
                                    (PROF(N1,N2,N3)) (A)
                                    (' NOW TEACHES THAT COURSE') (A);
 424     3           NEW_PROF:
                         PUT SKIP(2) EDIT(' INSERT THE NAME OF PROFESSOR YOU ') (A)
                                    ('NOW WISH TO TEACH THIS COURSE') (A);
 425     3               PUT SKIP;
```

```
          MAIN:PROCEDURE OPTIONS(MAIN);                                                                  PAGE    12

 STMT LEVEL NEST

 426     3                    READ FILE(IN_FILE) INTO(CARDS);
 427     3                    CALL R_READ(CARDS,WORDS);
 428     3                    N_PROF = WORDS;

 429     3                    DO J = 1 TO #_PROFS;
 430     3     1              IF PROFESSOR(J) = N_PROF THEN GO TO OUT;
 432     3     1              END;

 433     3             PUT SKIP EDIT('THIS PROFESSOR DOES NOT EXIST') (A);
 434     3                     GO TO NEW_PROF;

 435     3              OUT:  DO L = 1 TO #_OF_CLASSES(J,L2);
 436     3     1                  IF CLASS_TIME(J,L,L2) =
 437     3     1                      TIME(N1,N2,N3) THEN DO;
 438     3     2                          PUT EDIT('PROFESSOR ') (A)
                                                  (PROF(N1,N2,N3)) (A)
                                                  (' TEACHES ') (A)
                                                  (NAME(N1,N2,N3)) (SKIP,A)
                                                  (' SECTION ') (A)
                                                  (SECTION(N1,N2,N3)) (A)
                                                  (' AT THIS TIME') (A);
 439     3     2                          PUT EDIT('DOU YOU WISH TO TRY AGAIN') (A)
                                                  ('1=YES,2=NO') (A);
 440     3     2                          CALL ANSWER;
 441     3     2                          IF IALPHA = 1 THEN GO TO FOUND;
 443     3     2                          RETURN;
 444     3     2                          END;
 445     3     1                  END;

 446     3                    DO L = 1 TO #_OF_CLASSES(L1,L2);
 447     3     1              IF CLASS_NAME(L1,L,L2) = C_NAME THEN GO TO CHNG;
 449     3     1              END;

 450     3               CHNG:
                              DUMMY = CLASSES(L1,L,L2);
 451     3                    CLASSES(L1,L,L2) = CLASSES(L1,#_OF_CLASSES(L1,L2),L2);
 452     3                    #_OF_CLASSES(L1,L2) = #_OF_CLASSES(L1,L2) -1;
 453     3                    PROF(N1,N2,N3) = N_PROF;
 454     3                    L1 = J;
 455     3                    #_OF_CLASSES(L1,L2) = #_OF_CLASSES(L1,L2) + 1;
 456     3                    CLASSES(L1,#_OF_CLASSES(L1,L2),L2) = DUMMY;
 457     3                     PUT SKIP EDIT('THE PROFESSOR HAS BEEN REASSIGNED') (A);
 458     3                     RETURN;
 459     3                     END REASSGN;
 460     2                R_READ:  PROCEDURE(CARDS,WORDS);
 461     3                         DCL CARDS CHAR(80) VARYING,
                                       WORDS CHAR(20) VARYING,
                                       CARD2 CHAR(80),
                                       CARD1(80) CHAR(1) DEFINED CARD2,
                                       (FCHAR,LCHAR,ECHAR,I) FIXED BIN(15);
 462     3                         CARD2 = '';
 463     3                         CARD2 = CARDS;
 464     3                         ECHAR = LENGTH(CARDS);
 465     3                         DO I=1 TO ECHAR;
```

```
        MAIN:PROCEDURE OPTIONS(MAIN);                                                    PAGE    13

 STMT LEVEL NEST

  466    3    1              IF CARD1(I) ¬= ' ' THEN GO TO FOUND;
  468    3    1           END;
  469    3                WORDS = '';   CARDS = '';
  471    3                RETURN;
  472    3          FOUND:
                          FCHAR = I;
  473    3                IF CARD1(I) = ',' THEN DO; WORDS = '';
  476    3    1                          CARDS =SUBSTR(CARD2,I+1,ECHAR-I);
  477    3    1                                  RETURN;
  478    3    1                               END;
  479    3                LCHAR = INDEX(CARDS,',');
  480    3                IF LCHAR = 0 THEN DO;
  482    3    1                  WORDS =SUBSTR(CARDS,FCHAR,ECHAR-FCHAR+1);
  483    3    1                                 CARDS = '';
  484    3    1                                 RETURN;
  485    3    1                             END;
  486    3                     ELSE DO; WORDS = SUBSTR(CARDS,FCHAR,LCHAR-FCHAR);
  488    3    1                           CARDS = SUBSTR(CARD2,LCHAR+1,ECHAR-LCHAR);
  489    3    1                           RETURN;
  490    3    1                      END;
  491    3                END R_READ;
  492    2        CHANGE: PROCEDURE;
  493    3          DCL CONS CHAR(8),
                         (II,CSW) FIXED BIN(15);
  494    3           CSW = 0;
  495    3           ON CONVERSION BEGIN;
  497    4                         GO TO L1;
  498    4                         END;
  499    3       L1:PUT SKIP EDIT ('INSERT COURSE NAME AND SECTION ',
                                   'SEPARATED BY A COMMA')(A,A);
  500    3          PUT SKIP;
  501    3          READ FILE(IN_FILE) INTO(CARDS);
  502    3          CALL R_READ(CARDS,WORDS);
  503    3          C_NAME = WORDS;
  504    3          CALL R_READ(CARDS,WORDS);
  505    3          C_SEC = WORDS;
  506    3          CONS =C_NAME||C_SEC;
  507    3             DO I = 1 TO VC_#;
  508    3    1        IF CONS = TITLE(I) THEN GO TO L2;
  510    3    1        END;
  511    3          PUT SKIP EDIT ('THIS COURSE AND/OR SECTION ',
                                   'DOES NOT EXIST')(A,A);
  512    3          GO TO L1;
  513    3       L2: PUT SKIP EDIT('INSERT TERM,DAYS AND TIME ',
                                   'YOU WISH TO MOVE THE COURSE TO ',
                                   'ALL SEPARATED BY COMMAS  ')(A,A,A);
  514    3             PUT SKIP;
  515    3           READ FILE(IN_FILE) INTO(CARDS);
  516    3           CALL R_READ(CARDS,WORDS);
  517    3           C_TERM = WORDS;
  518    3           CALL R_READ(CARDS,WORDS);
  519    3           C_DAY = WORDS;
  520    3           CALL R_READ(CARDS,WORDS);
  521    3           BEGIN_TIME = WORDS;
```

```
STMT LEVEL NEST

 522     3              CALL R_READ(CARDS,WORDS);
 523     3              END_TIME = WORDS;
 524     3                IF C_TERM < 1 | C_TERM > 2 THEN GO TO L4;
 526     3              IF END_TIME < BEGIN_TIME THEN GO TO L4;
 528     3              CALL TIME_PERIOD(JUMP,K);
 529     3              IF JUMP = 0 THEN GO TO G1;
 531     3          L4: PUT SKIP EDIT('INVALID TERM/DAYS/TIMES, TRY AGAIN') (A);
 532     3                            GO TO L2;
 533     3           G1:N1=PROF_CODE(P1(I),P2(I),P3(I));
 534     3              IF #_OF_CLASSES(N1,C_TERM) = 0 THEN GO TO L6;
 536     3              PUT SKIP EDIT('FOLLOWING IS A LIST OF ',
                                     'COURSES TAUGHT ',
                                    'BY PROFESSOR ',PROFESSOR(N1)) (A,A,A,A);
 537     3                   DO F= 1 TO #_OF_CLASSES(N1,C_TERM);
 538     3    1              PUT SKIP EDIT(CLASS_NAME(N1,F,C_TERM)) (A(6))
                                           (CLASS_SECTION(N1,F,C_TERM)) (X(2),A(2))
                                           (CLASS_TIME(N1,F,C_TERM)) (X(2),A(14));

 539     3    1              END;
 540     3         L6: IF NO_CLASS(K,C_TERM) = 0 THEN GO TO F1;
 542     3                         PUT SKIP EDIT('FOLLOWING IS A LIST OF ',
                                                'ALL COURSES TAUGTH AT THE TIME ',
                                                'YOU REQUESTED')
                                               (A,A,A,A);
 543     3               PUT SKIP;
 544     3             DO J = 1 TO NO_CLASS(K,C_TERM);
 545     3    1        PUT EDIT(NAME(K,J,C_TERM),SECTION(K,J,C_TERM))
                                (SKIP,COL(10),A(6),X(5),A(2));
 546     3    1        END;
 547     3             IF NO_CLASS(K,C_TERM) < MAX_CLASS THEN GO TO F1;
 549     3             PUT SKIP EDIT('TOO MANY CLASSES') (A);
 550     3           F1:M = C_TERM;
 551     3              II = K;
 552     3              X2 = NO_CLASS(K,C_TERM);
 553     3              C_CODE = CODE(K,X2,M);
 554     3              C_SEQ_NO = SEQ_NO(K,X2,M);
 555     3              CALL CONFL(K,X2,M);
 556     3           F2:
                PUT SKIP EDIT('TYPE 1 - CHANGE ANOTHER COURSE',
                              '     2 - CHANGE THE SAME COURSE',
                              '     3 - PERFORM THE CHANGE',
                              '     4 - RETURN.') (A,SKIP,A,SKIP,A,SKIP,A);
 557     3              CALL ANSWER;
 558     3              IF IALPHA = 1 THEN GO TO L1;
 560     3              IF IALPHA = 2 THEN GO TO L2;
 562     3              IF IALPHA = 3 THEN GO TO L3;
 564     3                          ELSE GO TO L7;
 565     3           L3:
                N_PROF = PROFESSOR(N1);
 566     3              III = 0;
 567     3              CALL INSERT(III);
 568     3              CALL DELETE(I,CSW);
 569     3               III = 0;
 570     3              GO TO F2;
```

```
STMT LEVEL NEST

 571    3           L7:  IF CSW = 0 THEN GO TO L5;
 573    3                  CALL SORT_TABLE;
 574    3                  VC_# = VC_# - CSW;
 575    3                L5: III = 1;
 576    3                  RETURN;
 577    3                END CHANGE;
 578    2              INSERT: PROCEDURE(III);
 579    3                    DCL (J,III,LM,L,I)  FIXED BIN(15);
 580    3                  IF III = 0 THEN GO TO CHECK_PROF;
 582    3                  ON CONVERSION BEGIN; GO TO L1;  END;

 586    3           FOUND:   PUT SKIP EDIT('PLEASE TYPE COURSE NAME,SECTION, TERM')(A)
                                     (',DAY,BEGIN TIME,END TIME,PROF,EXPECTED ')(A)
                                     ('ENROLLMENT, AND SEQUENCE NUMBER')(SKIP,A)
                                     (' AND COURSE CODE:  ALL SEPARATED ')(A)
                                     (' BY COMMAS')(A);
 587    3                L1:  PUT SKIP;
 588    3                    READ FILE(IN_FILE) INTO(CARDS);
 589    3                    CALL R_READ(CARDS,WORDS);
 590    3                    C_NAME = WORDS;
 591    3                    CALL R_READ(CARDS,WORDS);
 592    3                    C_SEC = WORDS;
 593    3                    CALL R_READ(CARDS,WORDS);
 594    3                    C_TERM =WORDS;
 595    3                    CALL R_READ(CARDS,WORDS);
 596    3                    C_DAY = WORDS;
 597    3                    CALL R_READ(CARDS,WORDS);
 598    3                    BEGIN_TIME = WORDS;
 599    3                    CALL R_READ(CARDS,WORDS);
 600    3                    END_TIME = WORDS;
 601    3                    CALL R_READ(CARDS,WORDS);
 602    3                    N_PROF = WORDS;
 603    3                    CALL R_READ(CARDS,WORDS);
 604    3                    C_ENROLLMENT = WORDS;
 605    3                    CALL R_READ(CARDS,WORDS);
 606    3                    C_SEQ_NO = WORDS;
 607    3                    CALL R_READ(CARDS,WORDS);
 608    3                    C_CODE = WORDS;
 609    3              CHECK_PROF:
                             DO C_PROF_CODE = 1 TO #_PROFS;
 610    3     1                     IF PROFESSOR(C_PROF_CODE) = N_PROF THEN GO TO OUT;
 612    3     1                     END;

 613    3                    PUT SKIP EDIT('THAT PROFESSOR DOES NOT EXIST')(A)
                                     (' TRY AGAIN') (A);
 614    3                    PUT SKIP;
 615    3                    READ FILE(IN_FILE) INTO(CARDS);
 616    3                    CALL R_READ(CARDS,WORDS);
 617    3                    N_PROF = WORDS;
 618    3                    GO TO CHECK_PROF;
 619    3             OUT:   CALL TIME_PERIOD(JUMP,K);
 620    3                 IF JUMP = 0 THEN GO TO FILE_DAY;
 622    3                         ELSE DO;
 623    3     1                         PUT SKIP EDIT('INVALID INPUT') (A);
```

```
STMT LEVEL NEST

 624    3    1                          GO TO FOUND;
 625    3    1                          END;

 626    3          FILE_DAY:
 627    3                  IF NO_CLASS(K,C_TERM) +1 > MAX_CLASS THEN GO TO FIVE;
 628    3              M1 = C_PROF_CODE;
 629    3              F = #_OF_CLASSES(C_PROF_CODE,C_TERM);
 630    3              DO T = 1 TO F;
 631    3    1         K100 = CLASS_DATE(M1,T,C_TERM);
 632    3    1         IF K ¬= CLASS_PER(M1,T,C_TERM) THEN GO TO L_END;
 634    3    1         IF IC = K100 THEN GO TO SIX;
 636    3    1         IF IC > 7 THEN GO TO TT;
 638    3    1         IF IC = 1 | K100 = 1 THEN GO TO SIX;
 640    3    1         K200 = IC + K100;
 641    3    1         IF K200 = 9 THEN GO TO L_END;
 643    3    1         IF IC < 5 & K200 < 11 THEN GO TO SIX;
 645    3    1                              ELSE GO TO L_END;
 646    3    1      TT: IF K100 = 8 | IC = 8 THEN GO TO SIX;
 648    3    1         IF BEGIN_TIME > E_K(M1,T,C_TERM) THEN GO TO L_END;
 650    3    1         IF END_TIME < S_K(M1,T,C_TERM)  THEN GO TO L_END;
 652    3    1                                         ELSE GO TO SIX;
 653    3    1         L_END:END;
 654    3              M = C_TERM;
 655    3              X2 = NO_CLASS(K,M);
 656    3              CALL CONFL(K,X2,M);
 657    3              GO TO NC_CONF;
 658    3           NEXT_SEC:
 659    3              IF III = 0 THEN RETURN;
 660    3              PUT SKIP EDIT('TYPE 1 - INSERT ANOTHER COURSE',
                                    '      2 - RETURN') (A,SKIP,A);
 661    3                PUT SKIP;
 662    3                CALL ANSWER;
 663    3                IF IALPHA = 1 THEN GO TO FOUND;
 665    3                ELSE DO; CALL SORT_TABLE;
 667    3    1                RETURN;
 668    3    1                END;

 669    3           NO_CONF:
                       CALL FILE_PROF;
 670    3              GO TO NEXT_SEC;
 671    3          FIVE: PUT SKIP EDIT('TOO MANY CLASSES IN THIS PERIOD') (A)
                           (BEGIN_TIME) (A)
                           ('.  DO YOU STILL WISH TO ASSIGN IT TO THIS TIME?')(A)
                           ('1=YES,2=NO') (A);
 672    3              IF III = 0 THEN GO TO NC_CONF;
 674    3              PUT SKIP;
 675    3              CALL ANSWER;
 676    3              IF IALPHA = 1 THEN GO TO NO_CONF;
 678    3              GO TO FOUND;
 679    3          SIX: PUT SKIP EDIT('PROFESSOR  ')(A)
                           (PROFESSOR(C_PROF_CODE))  (A)
                           (' IS ALREADY TEACHING AT TIME')(A)
                           (BEGIN_TIME)(A);
 680    3              PUT SKIP;
```

```
        MAIN:PROCEDURE OPTIONS(MAIN);                                                    PAGE    17

STMT LEVEL NEST

 681    3           FINIS:  END INSERT;
 682    2             FILE_PROF: PROCEDURE;
 683    3                  NO_CLASS(K,C_TERM) = NC_CLASS(K,C_TERM) + 1;
 684    3                FILF_DAY:
                           AA = NO_CLASS(K,C_TERM);
 685    3                  NAME(K,AA,C_TERM) =C_NAMF;
 686    3                  SECTION(K,AA,C_TERM) = C_SEC;
 687    3                 T_DATE(K,AA,C_TERM) = IC;
 688    3                 T_PER(K,AA,C_TERM) = K;
 689    3                  CODE(K,AA,C_TERM) = C_CODE;
 690    3                  PROF(K,AA,C_TERM) = PROFESSOR(C_PROF_CODE);
 691    3                  SEQ_NC(K,AA,C_TERM) = C_SEQ_NO;
 692    3                   TRM(K,AA,C_TERM)=C_TERM;
 693    3                  ENROLLMENT(K,AA,C_TERM) = C_ENROLLMENT;
 694    3                  CLS_CODE(K,AA,C_TERM) = WC(IC);
 695    3                   PUT STRING(TIME(K,AA,C_TERM) ) EDIT
                            (C_DAY,'  ',BEGIN_TIME,'  ',END_TIME)(A(3),A(2),F(4),A(1),F(4));

 696    3                 PROF_CODE(K,AA,C_TERM) = C_PRCF_CODE;
 697    3                   IF AA > MAX_CLASS THEN
 698    3                      PUT EDIT('MORE THAN ')(A)
                                   (MAX_CLASS)(F(3))
                                   (' CLASSES HAVE BEEN SCHEDULED IN TIME PERIOD')(A)
                                   (TIME(K,AA,C_TERM))(A);
 699    3                   #_OF_CLASSES(C_PROF_CODE,C_TERM) =
                                 #_OF_CLASSES(C_PRCF_CODE,C_TERM) + 1;
 700    3                     B1 = #_OF_CLASSES(C_PROF_CODE,C_TERM);
 701    3                   CLASS_NAME(C_PROF_CODE,B1,C_TERM) = C_NAME;
 702    3                   CLASS_SECTION(C_PROF_CODE ,B1,C_TERM) = C_SEC;
 703    3                   CLASS_TIME(C_PROF_CODE,B1,C_TERM) = TIME(K,AA,C_TERM);
 704    3                 CLASS_DATE(C_PROF_CODE,B1,C_TERM) = IC;
 705    3                 CLASS_PER(C_PROF_CODE,B1,C_TERM) = K;
 706    3                         S_K(C_PROF_CODE,B1,C_TERM) = BEGIN_TIME;
 707    3                         E_K(C_PROF_CODE,B1,C_TERM) = END_TIME;
 708    3                   IF SW THEN GO TO SEC_ASS;
 710    3                   SW = '1'B;
 711    3                   VC_# = VC_# +1;
 712    3           TITLE(VC_#) = C_NAME || C_SEC;
 713    3                   P1(VC_#) = K;
 714    3                   P2(VC_#) = AA;
 715    3                   P3(VC_#) = C_TERM;

 716    3            SEC_ASS:
 717    3                   IF WC(IC) = 0 THEN GO TC FILE_NEXT;
 718    3                   ELSE DO;
 719    3    1                  IF K = 16 THEN GO TO FILE_NEXT;
 721    3    1                  L = (END_TIME - BEGIN_TIME)/100 -LM;
 722    3    1                  IF L = 0 THEN GO TO FILE_NEXT;
 724    3    1                  IF L = 1 & K > 9 THEN GO TO FILE_NEXT;
 726    3    1                  ELSE DO;
 727    3    2                       K = K + 1;
 728    3    2                       IF K = 10 THEN K = 16;
 730    3    2                       LM = LM + 1;
 731    3    2                       GO TO FILE_DAY;
```

```
STMT LEVEL NEST

 732     3    2                   END;
 733     3    1                END;
 734     3                     LM = 0;
 735     3           FILE_NEXT:
                        SW = '0'B;
 736     3              RETURN;
 737     3            END FILE_PROF;
 738     2          DELETE: PROCEDURE(I,CSW);
 739     3              DCL (I,CSW,J1,J2,J3,J4,J5,J6,L1,N1) FIXED BIN(15);
 740     3              J1 = P1(I);
 741     3              J2 = P2(I);
 742     3              J3 = P3(I);
 743     3              J5 = PROF_CODE(J1,J2,J3);
 744     3              N1 = PROF_CODE(J1,J2,J3);
 745     3              J4 = NO_CLASS(J1,J3);
 746     3              TIME_TABLE(J1,J2,J3) = TIME_TABLE(J1,J4,J3);
 747     3              NO_CLASS(J1,J3) = NO_CLASS(J1,J3) -1;
 748     3              J6 = #_OF_CLASSES(J5,J3);
 749     3              DO L1 = 1 TO J6;
 750     3    1         IF TITLE(I) = CLASS_NAME(J5,L1,J3)||CLASS_SECTION(J5,L1,J3)
 751     3    1                     THEN GO TO FOUND;
 752     3    1         END;
 753     3           FOUND: CLASSES(J5,L1,J3) = CLASSES(J5,J6,J3);
 754     3              TITLE(I) = 'ZZZZZZZZ';
 755     3              #_OF_CLASSES(J5,J3) = #_OF_CLASSES(J5,J3) -1;
 756     3              CSW = CSW + 1;
 757     3              RETURN;
 758     3              END DELETE;

 759     2           C_MATRIX: PROCEDURE;
 760     3              DCL (ROW,COL,ENR) FIXED BIN(15);
 761     3          L4: PUT SKIP EDIT('ENTER ROW,COLUMN AND ENROLLMENT ',
                                   '(RETURN IF ROW NUMBER = 0)') (A,SKIP,A);
 762     3          L3: PUT SKIP;
 763     3              READ FILE(IN_FILE) INTO(CARDS);
 764     3              CALL R_READ(CARDS,WORDS);
 765     3              ROW = WORDS;
 766     3              IF ROW = 0 THEN RETURN;
 768     3              IF ROW > #_COURSES THEN GO TO L1;
 770     3              CALL R_READ(CARDS,WORDS);
 771     3              COL = WORDS;
 772     3              IF COL < 1 | COL > #_COURSES THEN GO TO L1;
 774     3              CALL R_READ(CARDS,WORDS);
 775     3              ENR = WORDS;
 776     3              IF ENR < 1 THEN GO TO L1;
 778     3                         ELSE GO TO L2;
 779     3          L1: PUT SKIP EDIT('INVALID ROW/COL/ENROLLMENT, TRY AGAIN')(A);
 780     3              GO TO L3;
 781     3          L2: REQUEST_MATRIX(ROW,COL) = ENR;
 782     3              REQUEST_MATRIX(COL,ROW) = ENR;
 783     3              GO TO L4;
 784     3              END C_MATRIX;
 785     2            END MAIN;
```